Electronic Design Department
Poland, 44-100 Gliwice, Dubois 16
Phone/Fax: +48 32 2311171, 2313027
e-mail: ipcenter@evatronix.com
url: www.evatronix.com

April 15, 2011

R80251XC
Design Specification

Document Revision History

Revision	Date	Person	Changes made in document from previous revision
1.00	20.02.2008	T.T. P.P.	First version
1.01	10.12.2008	M.P.	Review & corrections.
1.02	29.12.2008	T.T. P.P.	Corrections
1.03	8.01.2009	T.T. P.P.	Correct subcomponents specification
1.04	13.02.2009	M.P.	Tracking on. OCDS modifications.
1.05	21.08.2009	M.P.	Tracking off. Cleanup.
1.06	21.05.2010	J.T	SFR's and Peripherals description corrections.
1.07	27.10.2010	M.P.	Added Register File access to the OCDS.
1.08	12.11.2010	M.P.	Corrected the Block Diagram and added 80C515-like Timer2 to the Features list.
1.09	15.11.2010	M.P. J.T.	Corrected UCONFIG0 and UCONFIG1 registers description. Corrected and complemented the PCA chapter.
1.10	10.01.2011	M.P. J.T	Corrected Pin Description table and External Memory / Internal Memory timing diagrams. Corrected and complemented the WATCHDOG chapter.
1.11	24.01.2011	M.P.	Modified Pin Description table, Feature list, Block Diagram and Interface Timings: R80251XC-I(F) version was added.
1.12	08.02.2011	M.P.	Review and cleanup.
1.13	15.04.2011	J.T.	Corrected ISR subcomponent, IEN2 and IRCON2 registers description.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Overview

1.1.1 Purpose Of Document

This document contains the design specifications of R80251XC IP core which is member of MCS-251 microcontroller family with additional components. This family of 8-bit microcontrollers is a high-performance upgrade of the widely-used MCS51 microcontrollers. The document contains architectural specification and hardware specification. Furthermore it details an overall component specification and provides functional description of all subcomponents.

1.1.2 Maintenance Of Document

The Design specification is developed in parallel to Verification Specification and Test Plan. All those documents are reviewed and signed-off before top-level testbench is coded.

1.2. Features

1.2.1 Features Of Reference Standard Implemented

a) Instruction Set

This IP core is compliant with the Intel MCS 251 instruction set.

b) I2C™ Serial Bus Interface

The R80251XC provides an interface to the Philips I2C™ serial bus.

c) Spi Serial Bus Interface

The R80251XC provides an interface to the Motorola SPI™ serial bus.

d) Other Peripherals

The R80251XC is equipped with peripherals functionally compatible with selected peripherals of Intel 80C251, Dallas 80C530 and Siemens SAB 80C515/80C517 microcontrollers:

- 80C251-like peripherals:
  - Timer 0, 1 & 2
  - Programmable Counter Array
  - Serial Port 0
  - 7-sources (extended to 21) Interrupt Controller with 4 priority levels
  - 8-bit Parallel I/O PORTS
  - Watchdog Timer
- SAB80C515-like peripherals:
  - Serial Port 1
  - Timer 2 with Compare / Capture Unit (CCU)
- Dallas 80C530-like peripherals:
  - Real Time Clock

1.2.2 Features Of Reference Standard Not Implemented

a) Other Peripherals

For the R80251XC-T(F) version, the 8-bit Parallel PORTS are not used as a multiplexed Program / Data Bus. All of them are general purpose I/O ports, which may be also off-core combined with dedicated pins of other peripherals.

1.2.3 Evatronix Proprietary Features

- Power Management Unit
- On-Chip Debug Support

1.2.4 List Of Features

- Control Unit
  - 8-bit Instruction decoder
  - Instructions operating on 8-bit, 16-bit, or 32-bit operands. (In comparison with 8-bit and 16-bit operands, 32-bit operands are accessed with fewer addressing modes)
- Arithmetic-Logic Unit
  - 32-bit arithmetic and logic instructions
  - Boolean manipulations
  - the MUL (multiply) and DIV (divide) instructions for unsigned 8-bit and 16-bit data
    - eight-bit multiplication: 8 bits × 8 bits -> 16 bits
    - sixteen-bit multiplication: 16 bits × 16 bits -> 32 bits
    - eight-bit division: 8 bits ÷ 8 bits -> 16 bits (8-bit quotient, 8-bit remainder)
    - sixteen-bit division: 16 bits ÷ 16 bits -> 32 bits (16-bit quotient, 16-bit remainder)
- Input/Output ports
  - Input/Output ports (optional)
    - Up to four 8-bit I/O ports
    - Alternate port functions such as external interrupts and serial interface are separated, providing extra port pins when compared with the standard 80251 (applies to R80251XC-T(F))
  - 16-bit Timers/Counters (optional)
    - 80C251-like Timer 0, 1
    - 80C251-like Timer 2
      - or
    - 80515-like Timer2 with CCU (applies to R80251XC-T(F))
  - Programmable Counter Array (optional)
    - 16-bit timer/counter
    - Five 16-bit capture/compare modules
    - Output signal generators
    - Pulse width modulators
    - Software Watchdog Timer
  - Full Duplex Serial Interfaces (optional)
    - Serial 0 (80C251-like)
      - Synchronous mode, fixed baud rate

        - 8-bit UART mode, variable baud rate
        - 9-bit UART mode, fixed baud rate
        - 9-bit UART mode, variable baud rate
    - Serial 1 (80517-like) (applies to R80251XC-T(F))
        - 8-bit UART mode, variable baud rate
        - 9-bit UART mode, variable baud rate
        - Baud Rate Generator
- Interrupt Controller
    - Four Priority Levels with 21 interrupt sources (80C251-like, with more sources)
- 14 bit Watchdog Timer (optional)
- Internal Data Memory interface
    - addresses up to 1024 B of Data Memory Space
- External Memory interface
    - addresses up to 16 MB of External Memory
    - De-multiplexed Address/Data Bus to ease the connection with memories (applies to R80251XC-T(F))
    - Compatible addressing with MCS-51 using DPTR or using DPX (extended data pointer) for addressing up to 16 MB
- Special Function Registers interface
    - Services to 128 External Special Function Registers (depending on peripheral configuration)
- Power Management Unit (optional)
    - Power down modes: IDLE and STOP
- Debug Interface for On-Chip Debug Support (OCDS) (optional)
- I2C interface (optional) (applies to R80251XC-T(F))
- Secondary I2C interface (optional) (applies to R80251XC-T(F))
- SPI interface (optional) (applies to R80251XC-T(F))
- Real Time Clock (optional) (applies to R80251XC-T(F))
- Extensive core configurability
    - external interrupts: 0 … 14 (applies to R80251XC-T(F))
    - external interrupts: 0 … 2 (applies to R80251XC-I(F))
    - Power management unit: 0 or 1
    - number of 8-bit I/O ports: 0 ... 4
    - number of 16-bit timers: 0, 1, 2 or 3
        - Timer 2 type: 80C251-like or 80C515-like (applies only to R80251XC-T(F))
    - Watchdog timer: 0 or 1
    - Programmable Counter Array: 0 or 1
    - number or serial ports: 0, 1 or 2 (applies to R80251XC-T(F))
    - number or serial ports: 0 or 1 (applies to R80251XC-I(F))
    - Software reset: 0 or 1 (applies to R80251XC-T(F))
    - I2C master-slave interface: 0 or 1 (applies to R80251XC-T(F))
    - Secondary I2C master-slave interface: 0 or 1 (only when basic I2C is implemented) (applies to R80251XC-T(F))
    - SPI master-slave interface: 0 or 1 (applies to R80251XC-T(F))

- Real Time Clock: 0 or 1 (applies to R80251XC-T(F))
- On-Chip Debug Support: 0 or 1
  - OCDS type (without Trace, with Program Trace, with Program and Data Trace): 0, 1 or 2
  - number of breakpoints: 2 … 8

NOTE: The 80C251-like Timer2 and PCA are mutually exclusive to the 80C515-like Timer2

1.3. Acceptance Criteria

The acceptance criteria for designed R80251XC component have been defined in the Verification Specification Test Plan for R80251XC component.

1.4. Terminology and Symbol Conventions

Table 1. Abbreviations and Acronyms

Symbol	Description
LSB	**L**east **S**ignificant **B**it
MSB	**M**ost **S**ignificant **B**it
SFR	**S**pecial **F**unction **R**egister
CPU	**C**ontrol **P**rocessor **U**nit
ALU	**A**rithmetic-**L**ogic **U**nit
ISR	**I**nterrupt **S**ervice **R**outine unit
PMU	**P**ower **M**anagement **U**nit
I2C	**I**nter-**IC** – a serial interface designed by Philips Semiconductors
SPI	**S**erial **P**eripheral **I**nterface
RTC	**R**eal **T**ime **C**lock
OCDS	**O**n-**C**hip **D**ebug **S**upport

Table 2. Terminology

Term	Description
Input	Input term when IP core pinout is described always means input to the core
Output	Output term when IP core pinout is described always means its output.

1.5. References

1.5.1 Industry Standards and Specifications

- MCS®51 Microcontroller Family User's Manual, Intel, February 1994
- 8-Bit CMOS Single-Chip Microcontroller SAB 80C515 / SAB 80C535, Siemens, February 1996
- 8-Bit CMOS Single-Chip Microcontroller SAB 80C517 / SAB 80C537, Siemens, April 1995
- High-Speed Microcontroller Data Book, Dallas Semiconductor, 1995
- TSC 80251 Programmer's Guide, Atmel, Rev E – December 2000
- 8XC251SA/SB/SP/SQ Embedded Microcontroller User's Manual, Intel, May 1996

- 8XC251SA/SB/SP/SQ High-Performance CHMOS Microcontroller – Commercial/Express, Intel, May 1996

1.5.2 Related Documents

- R80251XC – Verification Specification, Evatronix SA 2010
- R80251XC – Test Plan, Evatronix SA 2010
- R80251XC – Integration Manual, Evatronix SA 2010

1.5.3 Other Documents

- Reuse Methodology Manual, Second edition (RMM2), M. Keating, P. Bricaud, Kluwer Academic Publishers 1999

1.5.4 Web Sites

- http://www.evatronix.pl
- http://www.intel.com
- http://www.siemens.com
- http://www.atmel–wm.com

2. Architectural Specification

2.1. Pin Description

The pinout of the R80251XC has not been fixed to specific device I/O, thereby, allows flexibility with user application. The R80251XC contains only unidirectional pins - inputs or outputs. For proper communications via bi-directional PORTS 0-3, it is necessary to use in-circuit Open Drains.

Table 3. Pin Description

Name	Type	Polarity Bus size	Description
clkcpu	I	Rise	**Engine clock** Pulse for internal circuits, which are stopped when R80251XC is in IDLE or STOP mode
clkcpuen	O	High	**Engine clock enable output** External control for the "clkcpu" clock, when set to 1 the system clock should be applied to the "clkcpu" input, otherwise the "clkcpu" should be stopped
clkper	I	Rise	**Peripheral clock** Pulse for internal circuits, which are stopped when R80251XC is in STOP mode
clkperen	O	High	**Peripheral clock enable output** External control for the "clkper" clock, when set to 1 the system clock should be applied to the "clkper" input, otherwise the "clkper" should be stopped
reset	I	High	**Hardware reset input** High level on this pin for two clock cycles while both "clkcpu" and "clkper" are running resets the device
ro	O	High	**Reset output** Set active when either external reset, watchdog timer, software reset, or OCDS generates reset signal to the core
rtcx	I	Rise/Fall	**RTC 32,768kHz clock input (optional)** Used to trigger the RTC counters
rtcreset	I	High	**RTC Reset input (optional)** Used to reset the RTC (both synchronous and asynchronous way)
Port 0			
port0i port0o	I O	8 8	8-bit bi-directional I/O port with separated inputs and outputs
port0_strength*	O	High	When high, the Port0 drives strong 1. When low, Port0 should drive a pull-up.
port0_dir*	O	High	When high, the Port0 is configured as output driver. The strength of high state depends on 'port0_strength' output. The low state is always strong. When low, the Port0 is configured as input, no matter what values are written to 'port0o' or 'port0_strength'.
Port 1			
port1i port1o	I O	8 8	8-bit bi-directional I/O port with separated inputs and outputs
Port 2			

Name	Type	Polarity Bus size	Description
port2i port2o	I O	8 8	8-bit bi-directional I/O port with separated inputs and outputs
port2_strength*	O	High	When high, the Port2 drives strong 1. When low, Port2 should drive a pull-up.
Port 3			
port3i port3o	I O	8 8	8-bit bi-directional I/O port with separated inputs and outputs
port3_6_strength* port3_7_strength*	O O	High High	When high, the selected bits (6, 7) of Port3 are driven with strong 1. When low, Port3.6(7) should drive a pull-up.
External interrupts inputs			
int0	I	Low/Fall	External interrupt 0
int1	I	Low/Fall	External interrupt 1
int2	I	Fall/Rise	External interrupt 2
int3	I	Fall/Rise	External interrupt 3
int4	I	Rise	External interrupt 4
int5	I	Rise	External interrupt 5
int6	I	Rise	External interrupt 6
int7	I	Rise	External interrupt 7
int8	I	Rise	External interrupt 8
int9	I	Rise	External interrupt 9
int10	I	Rise	External interrupt 10
int11	I	Rise	External interrupt 11
int12	I	Rise	External interrupt 12
int13	I	Rise	External interrupt 13
Serial 0 interface			
rxd0i	I	-	Serial 0 receive data
rxd0o	O	-	Serial 0 transmit data
txd0	O	-	Serial 0 transmit data or receive clock in mode 0
Serial 1 interface			
rxd1	I	-	Serial 1 receive data
txd1	O	-	Serial 1 transmit data
I2C interface			
scli	I	-	Serial clock input
sdai	I	-	Serial data input
sclo	O	-	Serial clock output
sdao	O	-	Serial data output
Secondary I2C interface			
scl2i	I	-	Serial clock input
sda2i	I	-	Serial data input
scl2o	O	-	Serial clock output
sda2o	O	-	Serial data output
SPI interface			
scki	I	-	Serial clock input
scko	O	-	Serial clock output
scktri	O	High	Serial clock tri-state enable (to combine the "sck" bidirectional port)
ssn	I	Low	Slave select input
misoi	I	-	"Master input / slave output" input pin
misoo	O	-	"Master input / slave output" output pin
misotri	O	High	"Master input / slave output" tri-state enable (to combine the "miso" bidirectional port)

Name	Type	Polarity Bus size	Description
mosii	I	-	"Master output / slave input" input pin
mosio	O	-	"Master output / slave input" output pin
mositri	O	High	"Master output / slave input" tri-state enable (to combine the "mosi" bidirectional port)
spssn	O	8	Slave select output register
Timers interface			
t0	I	Fall	Timer 0 external input
t1	I	Fall	Timer 1 external input
t2i	I	Fall	Timer 2 external input
t2ex	I	Fall	Timer 2 capture trigger/count direction select
t2o	O	-	Timer 2 clock output
Programmable Counter Array			
eci	I	Fall	PCA external input
cexi(4:0)	I	Rise/Fall	PCA compare/capture inputs
cexo	O	5	PCA compare/capture outputs
External memory interface (R80251XC-I(F) version only)			
ea	I	Low	**External Access Input** Implemented only in R80251XC-I(F) version, used to enable the Code fetch from ROM interface ('ea'=1) or external memory (using Port0, Port2 and Port3) (when 'ea'=0).
ale	O	High	**Address Latch Enable Output** Driven high when the multiplexed bus composed of Port2 and Port0 provides valid address information to be latched outside of the core.
psen	O	Low	**Program Store Enable Output** Driven low when the External Program Memory is read.
External memory interface (R80251XC-T(F) version only)			
memack	I	High	Data Memory acknowledge
mempsack	I	High	Program Memory acknowledge
memdatai	I	8	Memory data input
memdatao	O	8	Memory data output
memaddr	O	24	Memory address
mempswr	O	High	Code Memory write enable
mempsrd	O	High	Code Memory read enable
memwr	O	High	Data Memory write enable
memrd	O	High	Data Memory read enable
External memory interface (for posedge-clocked memories) (R80251XC-T(F) version only)			
memdatao_comb	O	8	Memory data output
memaddr_comb	O	24	Memory address
mempswr_comb	O	High	Code Memory write enable
mempsrd_comb	O	High	Code Memory read enable
memwr_comb	O	High	Data Memory write enable
memrd_comb	O	High	Data Memory read enable
Internal Data Memory interface (for posedge-clocked memories)			
ramdatai	I	8	Data bus input
ramdatao	O	8	Data bus output
ramaddr	O	10	Data file address
ramwe	O	High	Data file write enable
ramoe	O	High	Data file output enable
External Special Function Registers interface			

Name	Type	Polarity Bus size	Description
sfrdatai	I	8	SFR data bus input
sfrdatao	O	8	SFR data bus output
sfraddr	O	7	SFR address
sfrwe	O	High	SFR write enable
sfroe	O	High	SFR output enable
sfrack	I	High	SFR acknowledge
Internal Program Memory Interface (R80251XC-I(F) version only)			
romdatai	I	8	On-chip Code Memory data input
romdatao	O	8	On-chip Code Memory data output
romaddr	O	ROMADDR _LENGTH	On-chip Code Memory address
romwr	O	High	On-chip Code Data Memory write enable
romrd	O	High	On-chip Code Data Memory read enable
On-Chip Debug Support interface (OCDS)			
trst	I	Low	Debug logic reset input (IEEE1149.1 Test Logic Reset)
tck	I	High	Debug clock (IEEE1149.1 Test Clock)
tms	I	High	Test Mode Select (IEEE1149.1 Test Mode Select)
tdi	I	High	Debug Data Input (IEEE1149.1 Test Data Input)
tdo	O	High	Debug Data Output (IEEE1149.1 Test Data Output)
tdoenable	O	High	Debug Data Output Enable
Trace RAM Interface (OCDS)**			
addr_buf0	O	BUF_SIZE	RAM0 address bus.
datao_buf0	O	65	RAM0 data output bus.
datai_buf0	I	65	RAM0 data input bus.
wr_buf0	O	High	RAM0 write enable signal.
rd_buf0	O	High	RAM0 read enable signal.
addr_buf1	O	BUF_SIZE	RAM1 address bus.
datao_buf1	O	65	RAM1 data output bus.
datai_buf1	I	65	RAM1 data input bus.
wr_buf1	O	High	RAM1 write enable signal.
rd_buf1	O	High	RAM1 read enable signal.

* Outputs existing only in R80251XC-I(F) versions.

** Implemented when Trace option of the OCDS is enabled.

2.2. Functional Description

The R80251XC is based on the 8XC251SX, which is the first member of the MCS 251 microcontroller family. This family of 8-bit microcontrollers is a high-performance upgrade of the widely-used MCS 51.

It provides software and hardware interrupts, interfaces for serial communication, timer system with compare-capture-reload resources, multi-purpose I/O ports, watchdog timer, real time clock and debugger interface.

The R80251XC is available in two CPU performance options:

- R80251XC-T(F) version, whose architecture eliminates redundant states and implements parallel execution of fetch and execution phases; the CPU state latency is shortened from 2 clock cycles to just one clock cycle in respect to the reference device
- R80251XC-I(F) version, which is timing-compatible to the 8XC251SX device from Intel

The R80251XC core architecture features:

- 24–bit linear addressing and up to 16 Mbytes of memory
- Register File based CPU with registers accessible as bytes, words and double words
- Instruction queue with the capability to configure the number of bytes in a queue (FIFO register-based memory)
- Enriched instruction set, including 32–bit arithmetic and logic instructions
- 64–Kbyte extended stack space
- Minimum instruction–execution time of one clock cycle
- Binary-code compatibility with MCS 51 microcontrollers.

2.3. Memory Organization

The R80251XC microcontroller has three address spaces: Memory Space, Special Function Register (SFR) space and Register File.

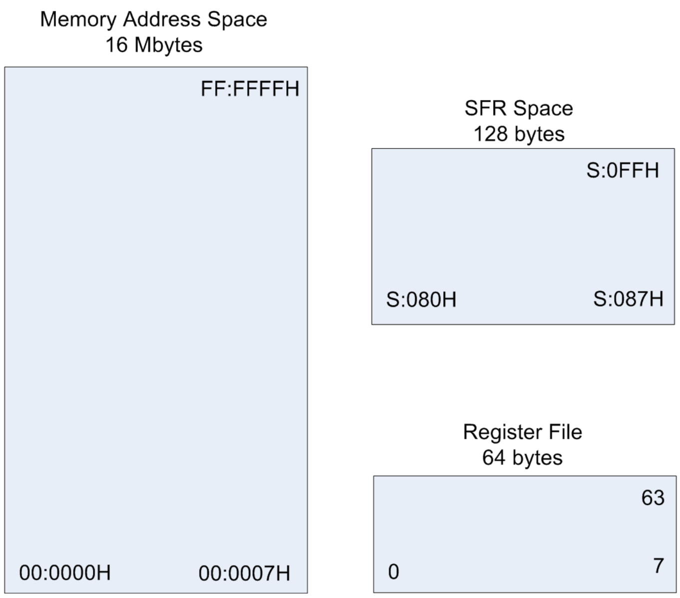


Figure 1. R80251XC Memory Map

The memory space in the MCS 251 architecture is un-segmented. The 64-Kbyte "regions" 00:, 01:, ..., FF: are introduced only as a convenience for discussions. Addressing in the MCS 251 architecture is linear; there are no segment registers.

2.3.1 Compatibility With the MCS 51 Architecture

The address spaces in the MCS 51 architecture are mapped into the address spaces in the MCS 251 architecture. This mapping allows code written for MCS 51 microcontrollers to run on MCS 251 microcontrollers.

Figure 2 shows the address spaces for the MCS 51 architecture. Internal data memory locations 00H–7FH can be addressed directly and indirectly. Internal data locations 80H–FFH can only be addressed indirectly. Directly addressing these locations accesses the SFRs. The 64-Kbyte code memory has a separate memory space. Data in the code memory can be accessed only with the MOVC instruction. Similarly, the 64-Kbyte external data memory can be accessed only with the MOVX instruction.

The register file (registers R0–R7) comprises of four switchable register banks, each having eight registers. The 32 bytes required for the four banks occupy locations 00H–1FH in the on-chip data memory.

Figure 3 shows how the address spaces in the MCS 51 architecture map into the address spaces in the MCS 251 architecture; details are listed inTable 4.

The 64-Kbyte code memory for MCS 51 microcontrollers maps into region FF: of the memory space for MCS 251 microcontrollers. Assemblers for MCS 251 microcontrollers assemble code for MCS 51 microcontrollers into region FF:, and data accesses to code memory are directed to this region. The assembler also maps the interrupt vectors to region FF:. This mapping is transparent to the user; code executes just as before, without modification.

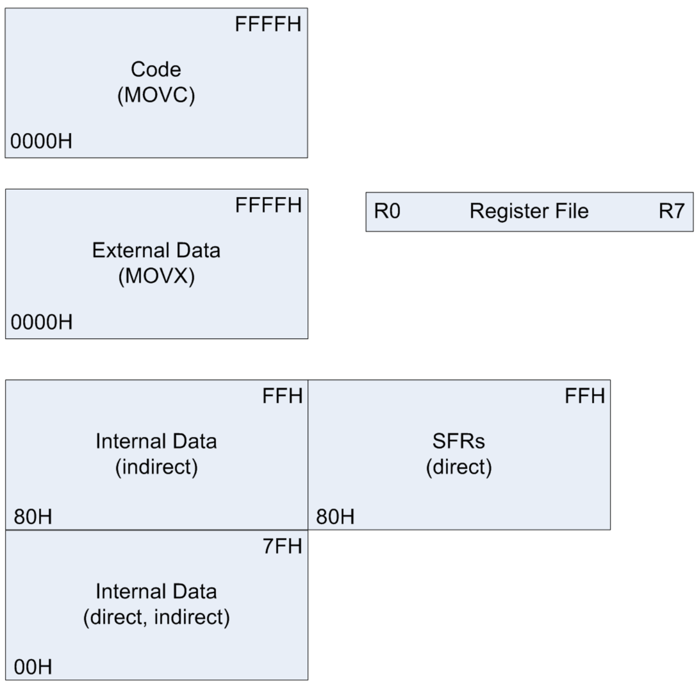


Figure 2. Address Spaces For the MCS 51 Architecture

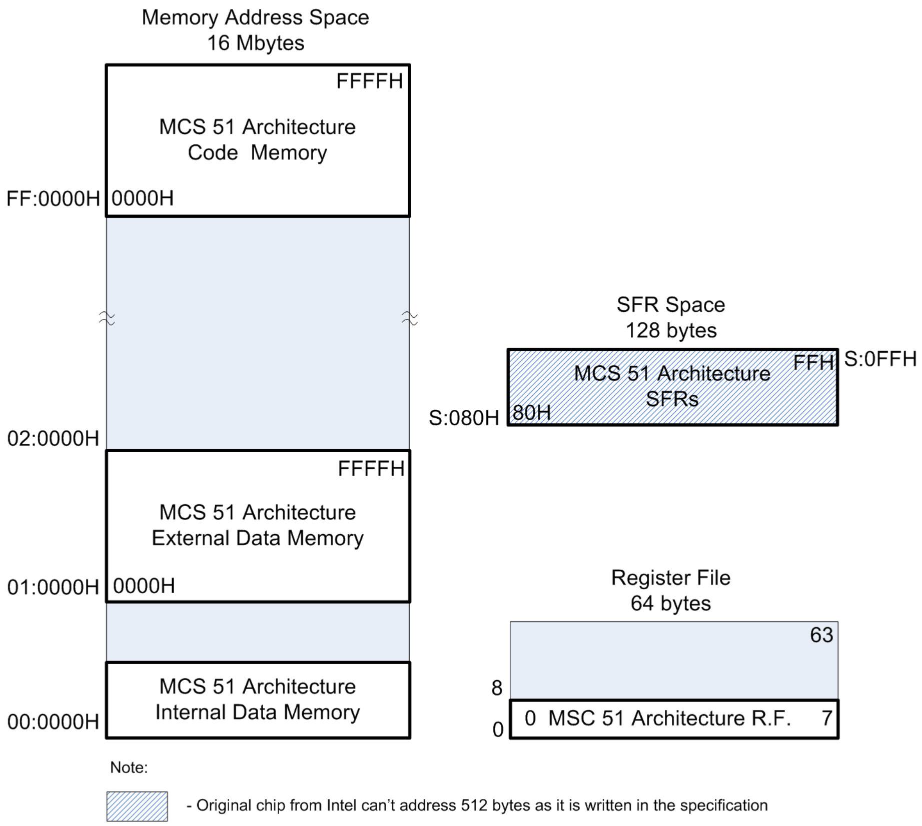


Figure 3. MCS-51 Address Space Mapping Into MCS-251 Architecture

Table 4. MCS-51 to MCS-251 Address Mapping

Memory Type	MCS 51 Architecture			MCS 251 Architecture
	Size	**Location**	**Data Addressing**	**Location**
Code	64 Kbytes	0000H–FFFFH	Indirect using MOVC instr.	FF:0000H–FF:FFFFH
External Data	64 Kbytes	0000H–FFFFH	Indirect using MOVX instr.	01:0000H–01:FFFFH
Internal Data	128 bytes	00H–7FH	Direct, Indirect	00:0000H–00:007FH
	128 bytes	80H–FFH	Indirect	00:0080H–00:00FFH
SFRs	128 bytes	S:80H–S:FFH	Direct	S:080H–S:0FFH
Register File	8 bytes	R0–R7	Register	R0–R7

The 64-Kbyte external data memory for MCS 51 microcontrollers is mapped into the memory region specified by bits 16–23 of the data pointer DPX, i.e., DPXL. DPXL is accessible as register file location 57 and also as the SFR at S:084H. The reset value of DPXL is 01H, which maps the external memory to region 01: as shown in Figure 3.

One can change this mapping by writing a different value to DPXL. A mapping of the MCS 51 microcontroller external data memory into any 64-Kbyte memory region in the MCS 251 architecture provides complete run-time compatibility because the lower 16 address bits are identical in the two address spaces.

The 256 bytes of on-chip data memory for MCS 51 microcontrollers (00H-FFH) are mapped to addresses 00:0000H-00:00FFH to ensure complete run-time compatibility. in the MCS 51

architecture, the lower 128 bytes (00H-7FH) are directly and indirectly addressable; however the upper 128 bytes are accessible by indirect addressing only. in the MCS 251 architecture, all locations in region 00: are accessible by direct, indirect, and displacement addressing.

The 128-byte SFR space for MCS 51 microcontrollers is mapped into the 128-byte SFR space of the MCS 251 architecture starting at address S:080H, as shown in Figure 3. This provides complete compatibility with direct addressing of MCS 51 microcontroller SFRs (including bit addressing). The SFR addresses are unchanged in the new architecture. in the MCS 251 architecture, SFRs A, B, DPL, DPH, and SP (as well as the new SFRs DPXL and SPH) reside in the register file for high performance. However, to maintain compatibility, they are also mapped into the SFR space at the same addresses as in the MCS 51 architecture.

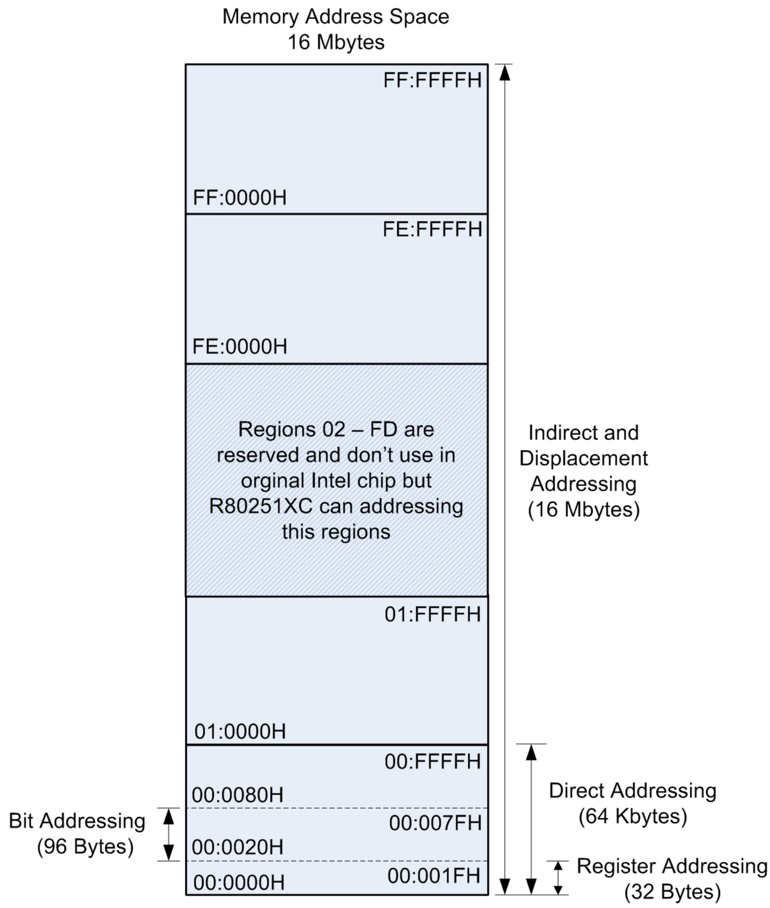


Figure 4. Logical Memory Address Space

2.3.2 Program and Code Memory

The External Bus Interface shares both Program Memory and External Data Memory and services memory when “memrd” or “memwr” signals are active.

Code can be executed only from external memory from 00:041FH to FF:FFF7H (FF:FFF8 – FF:FFFFFFH is configuration array). Code cannot be executed from Register File, SFRs and IRAM space (the code is only read from external memory using memrd signal). After reset the CPU reads Configuration Bytes from address FF:FFF8H and FF:FFF9H and after that code execution begins at FF:0000H (where MCS51 program memory is mapped). The lower part of program memory includes interrupt and reset vectors. The interrupt vectors are spaced at 8-byte intervals, starting from FF:0003h.
Call, Return and Jump instructions can be executed from whole external memory.

A variable length of fetch cycle and data instructions is introduced to access fast or slow ROM. Two bits of "uconfig0" register and two bits of "uconfig1" register together with external "memack" signal control memory wait states.

Two bits of "uconfig0" control number of wait states for all regions except 01: (see Table 53).

Two bits of "uconfig1" control number of wait states for region 01: (see Table 54).

If "memack" signal is asserted, the length of data instructions reading data from all regions depends on the settings of "uconfig1" or "uconfig1". If "memsack" is not acknowledged, wait state is inserted until "memack" is active again.

All data-related instructions read data according to the way of mapping shown in Figure 4 as follows:

- The first 32 bytes (00:0000H–00:001FH) provide storage for a part of the register file.
- Address range (00:0020H–00:041FH) is internal RAM, which is limited to 1kB, but when using direct addressing address range (00:0080H–00:00FFH) it is the SFR (see Figure 2 and Table 4).
- Other addresses refer to external memory.

2.3.3 Internal Data Memory

The R80251XC internal data memory interface services 1024 bytes of off-core data memory. The memory space accommodates also 128 bytes of Special Function Registers (see Figure 5).

Addresses lower than 80h access lower 128 bytes of internal data memory. Both direct or indirect addressing can be used in this case.

Indirect addressing of locations higher than 7Fh accesses upper 128 bytes of internal data memory, while direct addressing of locations higher than 7Fh accesses SFR space.

Addressing of locations higher than 0FFh accesses internal data memory.

The lower 128 bytes contain working registers (00h … 1Fh) and bit-addressable memory (20h … 2Fh). The lowest 32 bytes form four banks, each consisting of eight registers (R0-R7). Two bits of the program memory status word (PSW) select which bank is in use. The next 16 bytes of memory form a block of bit-addressable memory, accessible via 00h-7Fh addresses (see Figure 5).

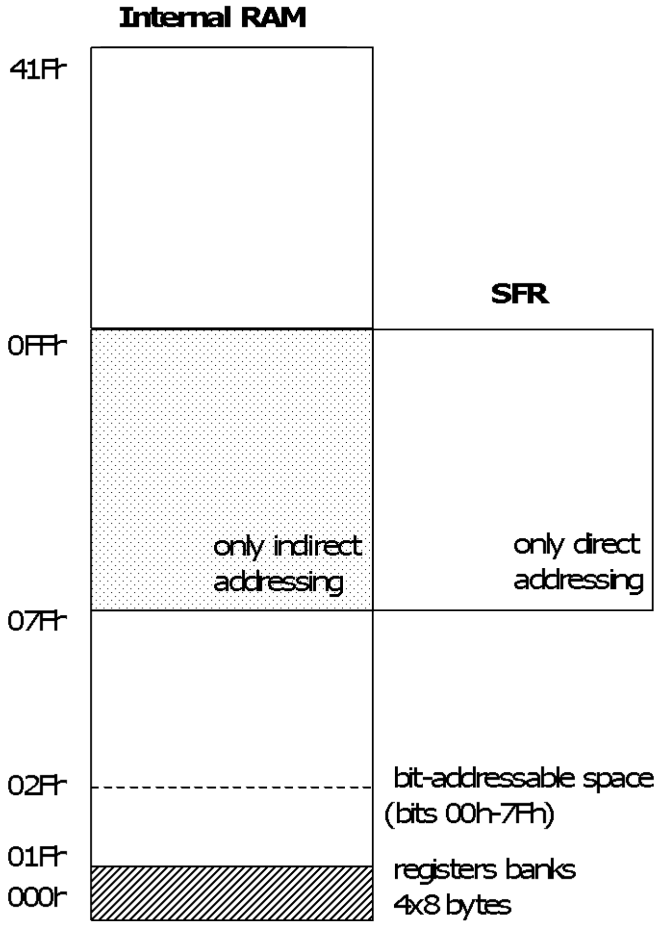


Figure 5. Internal Memory Map

2.3.4 The SFR Space

The SFR space can accommodate up to 128 8-bit special function registers with addresses in a range of S:080H–S:0FFH. Some of these locations may be unimplemented in a particular device. in the MCS 251 architecture, the prefix "S:" is used with SFR addresses to distinguish them from the memory space addresses 00:0000H–00:00FFH.

2.3.5 Register File

The register file has its own address space (Figure 1). The 64 locations in the register file are numbered decimally from 0 to 63. Locations 0–7 represent one of four switchable register banks, each having 8 registers. The 32 bytes required for these banks occupy locations 00:0000H–00:001FH in the memory space. Register file locations 8–63 do not appear in the memory space.

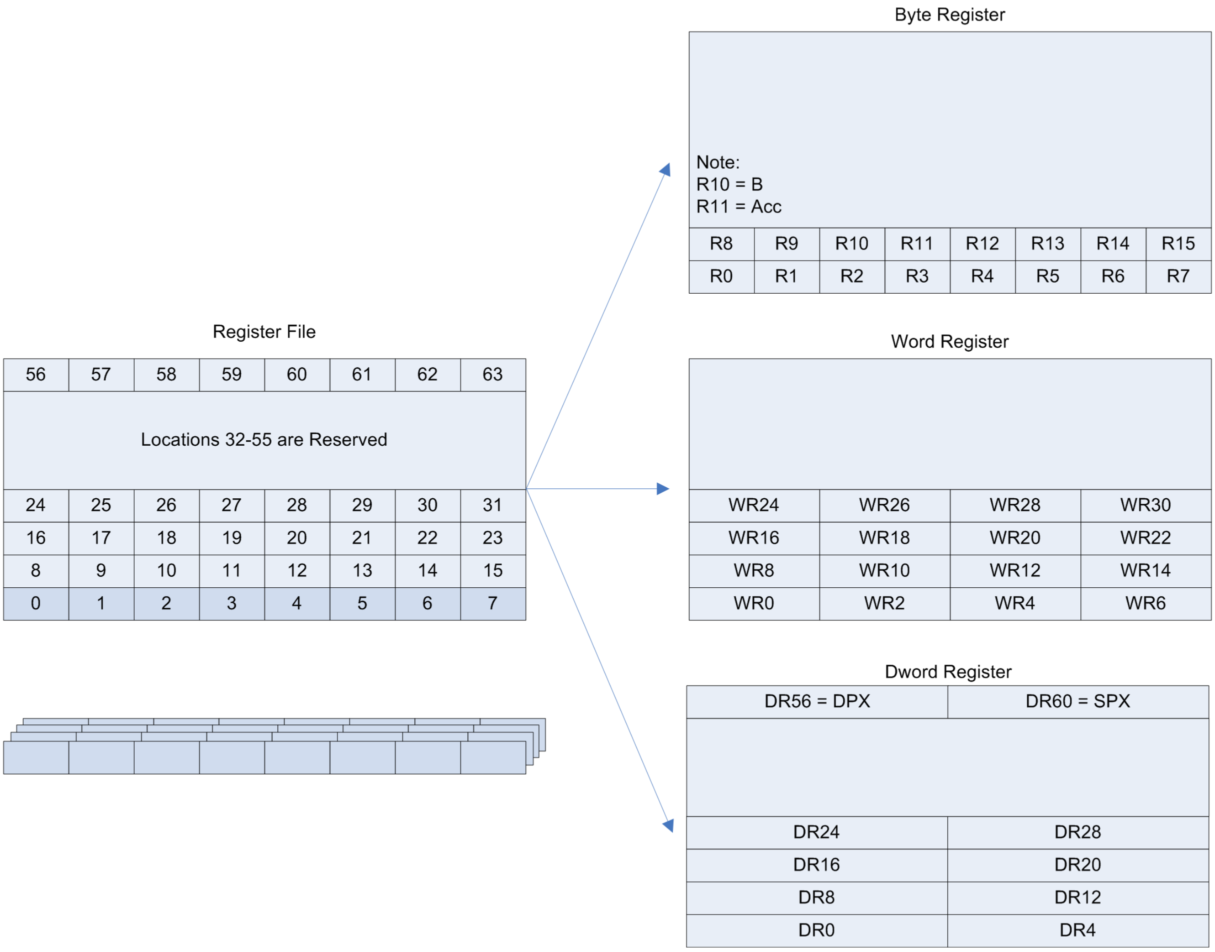


Figure 6. The Register File

Register file locations 0–7 actually consist of four switchable banks of eight registers each, as illustrated in Figure 7. The four banks are implemented as the first 32 bytes of internal RAM and are always accessible as locations 00:0000H–00:001FH in the memory address space. Only one of the four banks is accessible via the register file at a given time. The accessible, or "active," bank is selected by bits RS1 and RS0 in the PSW register, as shown in Table 5. Register Bank Selection. This bank selection can be used for fast context switches.

Register file locations 8–31 and 56–63 are always accessible. These locations are implemented as registers in the CPU. Register file locations 32–55 are reserved and cannot be accessed.

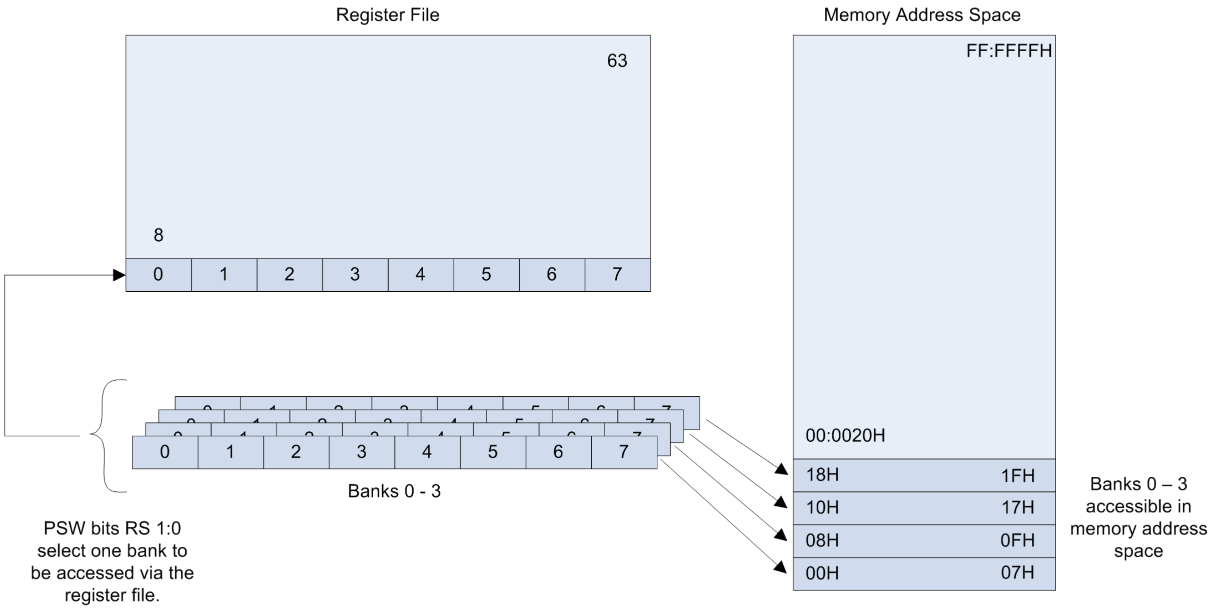


Figure 7. Register File Locations 0–7

Table 5. Register Bank Selection

Bank	Address range	PSW Selection Bits	
		RS1	RS0
Bank 0	00H - 07H	0	0
Bank 1	08H – 0FH	0	1
Bank 2	10H - 17H	1	0
Bank 3	18H – 1FH	1	1

2.3.6 Byte, Word and Dword Registers

Depending on its location in the register file, a register is addressable as a byte, a word, and/or a dword, as shown on the right side of Figure 6. A register is named for its lowest numbered byte location. For example:

- R4 is the byte register consisting of location 4.
- WR4 is the word register consisting of registers 4 and 5.
- DR4 is the dword register consisting of registers 4–7.

Locations R0–R15 are addressable as bytes, words, or dwords. Locations 16–31 are addressable only as words or dwords. Locations 56–63 are addressable only as dwords. Registers are addressed only by the names shown in Figure 6 - except for the 32 registers that comprise the four banks of registers R0–R7, which can also be accessed as locations 00:0000H–00:001FH in the memory space.

2.3.7 Dedicated Registers

The register file has four dedicated registers:

- R10 is the B-register
- R11 is the accumulator (ACC)
- DR56 is the extended data pointer, DPX
- DR60 is the extended stack pointer, SPX

These registers are located in the register file; however, R10, R11 and some bytes of DR56 and DR60 are also accessible as SFRs. The bytes of DPX and SPX can be accessed in the register file only by addressing the dword registers. The dedicated registers in the register file and their corresponding SFRs are illustrated in Figure 8 and listed in Table 6.

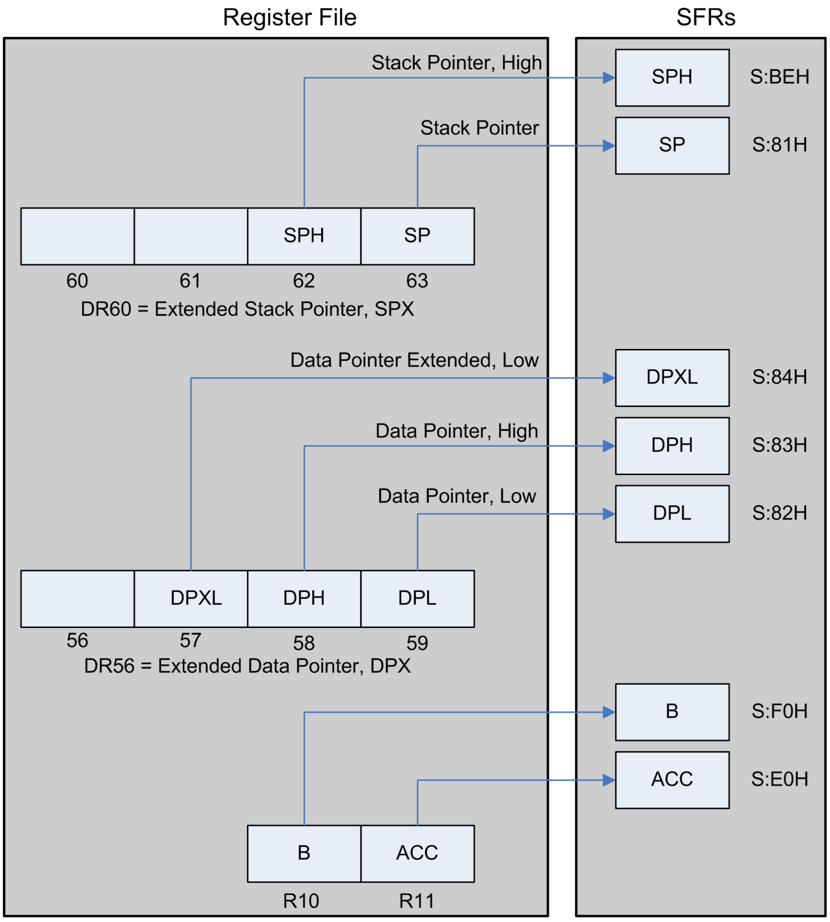


Figure 8. Dedicated Registers in the Register File and Corresponding SFRs

2.4. Special Function Registers

2.4.1 Special Function Registers Locations

The map of Special Function Registers is shown in Table 6. SFR addresses are preceded by "S:" to differentiate them from addresses in the memory space. Unoccupied locations in the SFR space are unimplemented, i.e., no register exists. If an instruction attempts to write to an unimplemented SFR location, the instruction executes, but nothing is actually written. If an unimplemented SFR location is read, it returns an unspecified value.

In the orginal Intel chip, SFRs may be accessed only as bytes; they may not be accessed as words or dwords. in R80251XC SFRs may be accessed as byte, word and dword.

Registers marked in gray are optional.

Table 6. Special Function Registers Locations

Hex/ Bin	X000	X001	X010	X011	X100	X101	X110	X111	Bin/ Hex
F8		ch	ccap0h	ccap1h	ccap2h	ccap3h	ccap4h		**FF**
F0	b		i2cdat	i2cadr	i2ccon	i2csta		srst	**F7**
E8	ien1	cl	ccap0l	ccap1l	ccap2l	ccap3l	ccap4l		**EF**
E0	acc	spsta	spcon	spdat	spssn				**E7**
D8	ccon	cmod	ccapm0	ccapm1	ccapm2	ccapm3	ccapm4		**DF**
D0	psw	psw1	i2c2dat	i2c2adr	i2c2con	i2c2sta			**D7**
C8	t2con	t2mod	rcap2l/ crcl	rcap2h/ crch	tl2	th2	rtcsel	rtcdata	**CF**
C0	ircon	ccen	ccl1	cch1	ccl2	cch2	ccl3	cch3	**C7**
B8	ipl0	s0aden		s1relh			sph	ircon2	**BF**
B0	p3							iph0	**B7**
A8	ien0	s0addr							**AF**
A0	p2						wdtrst	wcon	**A7**
98	s0con	s0buf	ien2	s1con	s1buf	s1rell			**9F**
90	p1	uconfig0	uconfig1						**97**
88	tcon	tmod	tl0	tl1	th0	th1			**8F**
80	p0	sp	dpl	dph	dpxl			pcon	**87**

In MCS-51:
The 16 addresses from SFR space are both byte- and bit-addressable. The bit-addressable SFRs are registers which addresses end with 000'b (80'h, 88'h, 90'h … F8'h). Those 16 registers (128 bits) together with 128 bits from internal data memory (locations 20'h … 2F'h) form the bit-addressable space (see Figure 4 and Table 4).

In MCS-251:
All SFRs from 80h to F8h are both byte- and bit-addressable.

2.4.2 Special Function Registers Reset Values

Table 7. Special Function Registers Reset Values

Register	Location	Reset value	Description
p0	80h	FFh	Port 0 Register

Register	Location	Reset value	Description
sp	81h	07h	Stack Pointer – LSB of SPX
dpl	82h	00h	Data Pointer – Low Byte
dph	83h	00h	Data Pointer – High Byte
dpxl	84h	01h	Data Pointer Extended Low
pcon	87h	00h	Power Control
tcon	88h	00h	Timer/Counter Control Register
tmod	89h	00h	Timer Mode Register
tl0	8Ah	00h	Timer 0 Register, low byte
tl1	8Bh	00h	Timer 1 Register, low byte
th0	8Ch	00h	Timer 0 Register, high byte
th1	8Dh	00h	Timer 1 Register, high byte
p1	90h	FFh	Port 1 Register
uconfig0	91h	FFh	Configuration byte 0
uconfig1	92h	FFh	Configuration byte 1
s0con	98h	00h	Serial Port 0, Control Register
s0buf	99h	00h	Serial Port 0, Data Buffer
ien2	9Ah	00h	Interrupt Enable Register 2
s1con	9Bh	00h	Serial Port 1, Control Register
s1buf	9Ch	00h	Serial Port 1, Data Buffer
s1rell	9Dh	00h	Serial Port 1, Reload Register, low byte
p2	A0h	FFh	Port 2 Register
wdrst	A6h	00h	Watchdog Reset Register
wcon	A7h	00h	Waitstate Control Register
ien0	A8h	00h	Interrupt Enable Register 0
s0addr	A9h	00h	Serial Port 0, Slave Address Register
p3	B0h	FFh	Port 3 Register
iph0	B7h	00h	Interrupt Priority Register 0, high byte
ipl0	B8h	00h	Interrupt Priority Register 0, low byte
s0aden	B9h	00h	Serial Port 0 Slave Address Mask Register
s1relh	BBh	03h	Serial Port 1, Reload Register, high byte
sph	BEh	00h	Stack Pointer High – MSB of SPX
ircon2	BFh	00h	Interrupt Request Control 2 Register
ircon	C0h	00h	Interrupt Request Control Register
ccen	C1h	00h	Timer 2 (515) Compare/Capture Enable Register
ccl1	C2h	00h	Timer 2 (515) Compare/Capture Register 1, low byte
cch1	C3h	00h	Timer 2 (515) Compare/Capture Register 1, high byte
ccl2	C4h	00h	Timer 2 (515) Compare/Capture Register 2, low byte
cch2	C5h	00h	Timer 2 (515) Compare/Capture Register 2, high byte

Register	Location	Reset value	Description
ccl3	C6h	00h	Timer 2 (515) Compare/Capture Register 3, low byte
cch3	C7h	00h	Timer 2 (515) Compare/Capture Register 3, high byte
t2con	C8h	00h	Timer 2 (515)/(251) Control Register
t2mod	C9h	00h	Timer 2 (251) Mode Control Register
rcap2l/crcl	CAh	00h	Timer 2 (251) Reload/Capture Register, low byte/ Timer 2 (515) Compare/Reload/Capture Register, low byte
rcap2h/crch	CBh	00h	Timer 2 (251) Reload/Capture Register, high byte/ Timer 2 (515) Compare/Reload/Capture Register, high byte
tl2	CCh	00h	Timer 2, low byte
th2	CDh	00h	Timer 2, high byte
rtcsel	CEh	00h	RTC Select Register (address port to RTC)
rtcdata	CFh	00h	RTC Data Register (data port to RTC)
psw	D0h	00h	Program Status Word
psw1	D1h	00h	Program Status Word 1
i2c2dat	D2h	00h	Secondary I2C Data Register
i2c2adr	D3h	00h	Secondary I2C Address Register
i2c2con	D4h	00h	Secondary I2C Control Register
i2c2sta	D5h	F8h	Secondary I2C Status Register
ccon	D8h	00h	PCA Control Register
cmod	D9h	00h	PCA Mode Register
ccapm0	DAh	00h	PCA Compare/Capture Module Mode Register 0
ccapm1	DBh	00h	PCA Compare/Capture Module Mode Register 1
ccapm2	DCh	00h	PCA Compare/Capture Module Mode Register 2
ccapm3	DDh	00h	PCA Compare/Capture Module Mode Register 3
ccapm4	DEh	00h	PCA Compare/Capture Module Mode Register 4
acc	E0h	00h	Accumulator
spsta	E1h	00h	Serial Peripheral Status Register
spcon	E2h	14h	Serial Peripheral Control Register
spdat	E3h	00h	Serial Peripheral Data Register
spssn	E4h	FFh	Serial Peripheral Slave Select Register
ien1	E8h	00h	Interrupt Enable Register 1
cl	E9h	00h	PCA timer/counter register, low byte
ccap0l	EAh	00h	PCA Compare/Capture Register 0, low byte
ccap1l	EBh	00h	PCA Compare/Capture Register 1, low byte
ccap2l	ECh	00h	PCA Compare/Capture Register 2, low byte
ccap3l	EDh	00h	PCA Compare/Capture Register 3, low byte
ccap4l	EEh	00h	PCA Compare/Capture Register 4, low byte
b	F0h	00h	B Register
i2cdat	F2h	00h	I2C Data Register

Register	Location	Reset value	Description
i2cadr	F3h	00h	I2C Address Register
i2ccon	F4h	00h	I2C Control Register
i2csta	F5h	F8h	I2C Status Register
srst	F7h	00h	Software Reset Register
ch	F9h	00h	PCA timer/counter register, high byte
ccap0h	FAh	00h	PCA Compare/Capture Register 0, high byte
ccap1h	FBh	00h	PCA Compare/Capture Register 1, high byte
ccap2h	FCh	00h	PCA Compare/Capture Register 2, high byte
ccap3h	FDh	00h	PCA Compare/Capture Register 3, high byte
ccap4h	FEh	00h	PCA Compare/Capture Register 4, high byte

2.4.3 ACC - Accumulator

Address – E0h
Reset value – 00h

The mnemonics for accumulator-specific instructions refer to accumulator as A, not ACC. The 8-bit accumulator (ACC) is byte register R11, which is also accessible in the SFR space as ACC at S:E0H. Accessing ACC as a register is one state faster than accessing them as SFRs.

Instructions in the MCS 51 architecture use the accumulator as the primary register for data moves and calculations. However, in the MCS 251 architecture, any of registers R0–R15 can serve for these tasks. As a result, the accumulator does not play the central role that it has in MCS 51 microcontrollers.

2.4.4 B - B Register

Address – F0h
Reset value – 00h

The B register, used in multiplies and divides, is register R10, which is also accessible in the SFR space as B at S:F0H. Accessing B as a register is one state faster than accessing them as SFRs.

2.4.5 CCAPnH, CCAPnL – PCA Compare/Capture Registers (n = 0 … 4)

CCAP4H:
Address – FEh
Reset value – 00h
optional register

CCAP4L:
Address – EEh
Reset value – 00h
optional register

CCAP3H:
Address – FDh
Reset value – 00h
optional register

CCAP3L:
Address – EDh
Reset value – 00h
optional register

CCAP2H:

Address – FCh

Reset value – 00h

optional register

CCAP1H:

Address – FBh

Reset value – 00h

optional register

CCAP0H:

Address – FAh

Reset value – 00h

optional register

CCAP2L:

Address – ECh

Reset value – 00h

optional register

CCAP1L:

Address – EBh

Reset value – 00h

optional register

CCAP0L:

Address – EAh

Reset value – 00h

optional register

Compare/Capture Registers are 16-bit registers used in the operation of Compare/Capture Unit associated with the PCA module (detailed description -).

CCAPnH holds higher byte and CCAPnL holds lower byte of the 16 bit comparison/captured value for the corresponding compare/capture modules. When the PWM mode is set this registers control the duty cycle of the output signal.

CCAP4H, CCAP4L, CCAP3H, CCAP3L, CCAP2H, CCAP2L, CCAP1H, CCAP1L, CCAP0H, CCAP0L registers are allocated in the SFR memory space when PCA module is selected during IP core configuration.

2.4.6 CCAPMn – PCA Compare/Capture Mode Registers (n = 0 … 4)

CCAPM4:

Address – DEh

Reset value – 00h

optional register

CCAPM3:

Address – DDh

Reset value – 00h

optional register

CCAPM2:

Address – DCh

Reset value – 00h

optional register

CCAPM1:

Address – DBh

Reset value – 00h

optional register

CCAPM0:

Address – DAh

Reset value – 00h

optional register

PCA Compare/Capture Module Mode Registers determine operating mode of the corresponding compare/capture module.

Table 8. CCAPMn Register

Bit	Symbol	Description	Type
ccapm*n*.7	-	not used, read as 0	R
ccapm*n*.6	ecom*n*	**Compare** When set enables the module comparator which is used for timer, signal output, pulse width modulation and software watchdog timer (fourth compare/capture module only).	R/W
ccapm*n*.5	capp*n*	**Capture on Positive Edge** When set enables the capture triggered on positive edge on the cexi*n* input.	R/W
ccapm*n*.4	cap*n*	**Capture on Negative Edge** When set enables the capture triggered on negative edge on the cexi*n* input.	R/W
ccapm*n*.3	mat*n*	**Match** When set, if the value of the PCA timer/counter and value of compare/capture registers are equal then the ccf*n* bit in the ccon register is set. When ecom*n* =1 and mat*n* = 1 the software timer mode is set.	R/W
ccapm*n*.2	tog*n*	**Toggle** When set, if the value of the PCA timer/counter and value of compare/capture registers are equal then the cexo*n* output is toggled. When ecom*n* =1 and mat*n* = 1 and tog*n* = 1 the high-speed output mode is set.	R/W
ccapm*n*.1	pwm*n*	**Pulse Width Modulation** When set, the module operates as 8-bit pulse width modulator which generates output waveform on the cexo*n* output.	R/W
ccapm*n*.0	eccf*n*	**Enable ccf*n* interrupt** When set enables an interrupt request when the ccf*n* flag in the ccon register is set.	R/W

CCAPM4, CCAPM3, CCAPM2, CCAPM1, CCAPM0 registers are allocated in the SFR memory space when PCA module is selected during IP core configuration.

2.4.7 CCEN - Timer 2 (515) Compare/Capture Enable Register

Address – C1h
Reset value – 00h
optional register

CCEN register serves as a configuration register for Compare/Capture Unit associated with the Timer 2 (515) (detailed description – Chapter 5.6).

CCEN register is allocated in the SFR memory space when Timer 2 (515) is selected during IP core configuration.

Table 9. CCEN Register

Bit	Symbol	Description			Type
ccen.7 ccen.6	cocah3 cocal3	compare/capture mode for CC3 register			R/W R/W
		cocah3	**cocal3**	**Description**	
		0	0	compare/capture disabled	
		0	1	capture on rising edge at pin cc0	
		1	0	compare enabled	
		1	1	capture on write operation into register cc3	
ccen.5 ccen.4	cocah2 cocal2	compare/capture mode for CC2 register			R/W R/W
		cocah2	**cocal2**	**Description**	
		0	0	compare/capture disabled	
		0	1	capture on rising edge at pin cc1	
		1	0	compare enabled	
		1	1	capture on write operation into register ccl2	
ccen.3 ccen.2	cocah1 cocal1	compare/capture mode for CC1 register			R/W R/W
		cocah1	**cocal1**	**Description**	
		0	0	compare/capture disabled	
		0	1	capture on rising edge at pin cc2	
		1	0	compare enabled	
		1	1	capture on write operation into register cc1	
ccen.1 ccen.0	cocah0 cocal0	compare/capture mode for CRC register			R/W R/W
		cocah0	**cocal0**	**Description**	
		0	0	compare/capture disabled	
		0	1	capture on falling/rising edge at pin cc3	
		1	0	compare enabled	
		1	1	capture on write operation into register crcl	

2.4.8 CCH1, CCL1, CCH2, CCL2, CCH3, CCL3 – Timer 2 (515) Compare/Capture Registers

CC1:

CCL1:
Address – C2h
Reset value – 00h
optional register

CCH1:
Address – C3h
Reset value – 00h
optional register

CC2:

CCL2:
Address – C4h
Reset value – 00h
optional register

CCH2:
Address – C5h
Reset value – 00h
optional register

CC3:

CCL3:			**CCH3:**		
Address	**–**	**C6h**	**Address**	**–**	**C7h**
Reset value	**–**	**00h**	**Reset value**	**–**	**00h**
optional register			**optional register**		

Compare/Capture Registers (CC1, CC2, CC3) are 16-bit registers used in the operation of Compare/Capture Unit associated with the Timer 2 (detailed description - 5.4).

CCHn holds higher byte and CCLn holds lower byte of the CCn register.

CCL1, CCH1, CCL2, CCH2, CCL3, CCH3 registers are allocated in the SFR memory space when Timer 2 (515) is selected during IP core configuration.

2.4.9 CRCH, CRCL – Timer 2 (515) Compare/Reload/Capture Register

CRCL:			**CRCH:**		
Address	**–**	**CAh**	**Address**	**–**	**CBh**
Reset value	**–**	**00h**	**Reset value**	**–**	**00h**
optional register			**optional register**		

Compare/Reload/Capture Register CRC is a 16-bit wide register used in the operation of Compare/Capture Unit associated with Timer 2 (detailed description - 5.4).

CRCH holds higher byte and CRCL holds lower byte.

CRCL and CRCH registers are allocated in the SFR memory space when Timer 2 (515) is selected during IP core configuration.

2.4.10 CCON – PCA Control Register

Address – D8h
Reset value – 00h
optional register

2.4.11 CH, CL – PCA timer/counter registers

CH:			**CL:**		
Address	**–**	**F9h**	**Address**	**–**	**E9h**
Reset value	**–**	**00h**	**Reset value**	**–**	**00h**
optional register			**optional register**		

2.4.12 CMOD - PCA Mode Register

Address – D9h
Reset value – 00h
optional register

2.4.13 DPXL, DPH, DPL - Data Pointer

DPXL:

Address – 84h

Reset value – 01h

DPH:

Address – 83h

Reset value – 00h

DPL:

Address – 82h

Reset value – 00h

Dword register DR56 is the extended data pointer, DPX. The lower three bytes of DPX (DPL, DPH, and DPXL) are accessible as SFRs. DPL and DPH comprise the 16-bit data pointer DPTR. While instructions in the MCS 51 architecture always use DPTR as the data pointer, instructions in the MCS 251 architecture can use any word or dword register as a data pointer. DPXL, the byte in location 57 in register file, specifies the region of memory (00:–FF:) that maps into the 64-Kbyte external data memory space in the MCS 51 architecture. in other words, the MOVX instruction addresses the region specified by DPXL when it moves data to and from external memory.

2.4.14 I2C2ADR - Secondary I2C Address Register

Address – D3h

Reset value – 00h

optional register

The I2C2ADR register register holds the own address of R80251XC Secondary I2C slave interface. This address is used to recognize if an external device is attempting to access the R85015X-C as slave via Secondary I2C bus.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

The I2C2ADR register is allocated in the SFR memory space when both I2C and Secondary I2C are selected during IP core configuration.

Table 10. I2C2ADR Register

Bit	Symbol	Description	Type
i2c2adr.7 i2c2adr.6 i2c2adr.5 i2c2adr.4 i2c2adr.3 i2c2adr.2 i2c2adr.1	adr	**Own Secondary I2C slave address** (7 bit)	R/W
i2c2adr.0	gc	**General Call Address Acknowledge** If this bit is set, the general call address is recognized; otherwise it is ignored.	R/W

2.4.15 I2C2CON - Secondary I2C Control Register

Address – D4h
Reset value – 00h
optional register

The I2C2CON register controls the operation of Secondary I2C interface.

The CPU can read from and write to this 8-bit, directly addressable SFR. Two bits of this register are affected by the Secondary I2C hardware: the **si** bit is set when serial interrupt is requested, and the "sto" bit is cleared when STOP condition is present on the Secondary I2C bus.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

The I2C2CON register is allocated in the SFR memory space when both I2C and Secondary I2C are selected during IP core configuration.

Table 11. I2C2CON Register

Bit	Symbol	Description	Type
i2c2con.7	cr2	**Clock rate bit 2**	R/W
i2c2con.6	ens1	**I2C enable bit** When ens1='0' the "sdao" and "sclo" outputs are set to 1, that drives the output pads of the chip in high impedance, and "sdai" and "scli" input signals are ignored. When ens1='1' I2C component is enabled.	R/W
i2c2con.5	sta	**START Flag** When sta='1', the I2C component checks the I2C bus status and if the bus is free a START condition is generated.	R/W
i2c2con.4	sto	**STOP Flag** When sto='1' and I2C interface is in master mode, a STOP condition is transmitted to the I2C bus.	R/W
i2c2con.3	si	**Serial Interrupt Flag** The "si "is set by hardware when one of 25 out of 26 possible I2C states is entered (2.4.17). The only state that does not set the "si" is state F8h, which indicates that no relevant state information is available. The "si" flag must be cleared by software. in order to clear the "si" bit, '0' must be written to this bit. Writing a '1' to si bit does not change value of the "si".	R/W

Bit	Symbol	Description	Type
i2c2con.2	aa	**Assert Acknowledge Flag** When aa='1', an "acknowledge" will be returned when: - the "own slave address" has been received - the general call address has been received while gc bit in i2caddr register was set - a data byte has been received while I2C was in master receiver mode - a data byte has been received while I2C was in slave receiver mode When aa='0', an "not acknowledge" will be returned when: - a data byte has been received while I2C was in master receiver mode - a data byte has been received while I2C was in slave receiver mode	R/W
i2c2con.1	cr1	**Clock rate bit 1**	R/W
i2c2con.0	cr0	**Clock rate bit 0**	R/W

For details on clock rate settings see section 5.17.4 g).

2.4.16 I2C2DAT - Secondary I2C Data Register

Address – D2h

Reset value – 00h

optional register

The I2C2DAT register contains a byte to be transmitted through Secondary I2C bus or a byte which has just been received through Secondary I2C bus. The CPU can read from and write to this 8-bit, directly addressable SFR while it is not in the process of byte shifting. The I2C2DAT register is not shadowed or double buffered so the user should only read I2C2DAT when an Secondary I2C interrupt occurs.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

The I2C2DAT register is allocated in the SFR memory space when both I2C and Secondary I2C are selected during IP core configuration.

2.4.17 I2C2STA - Secondary I2C Status Register

Address – D5h

Reset value – F8h

optional register

The contents of this register reflect the actual state of Secondary I2C interface.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

The I2C2STA register is allocated in the SFR memory space when both I2C and Secondary I2C are selected during IP core configuration.

Table 12. I2C2STA Register

Bit	Symbol	Description	Type
i2c2sta.7 i2c2sta.6 i2c2sta.5 i2c2sta.4 i2c2sta.3	-	**Secondary I2C Status Code**	R
i2c2sta.2 i2c2sta.1 i2c2sta.0	-	Not implemented, read as 0	R

2.4.18 I2CADR - I2C Address Register

Address – F3h
Reset value – 00h
optional register

The I2CADR register register holds the own address of R80251XC I2C slave interface. This address is used to recognize if an external device is attempting to access the R85015X-C as slave via I2C bus.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

I2CADR register is allocated in the SFR memory space when I2C is selected during IP core configuration.

Table 13. I2CADR Register

Bit	Symbol	Description	Type
i2cadr.7 i2cadr.6 i2cadr.5 i2cadr.4 i2cadr.3 i2cadr.2 i2cadr.1	adr	**Own I2C slave address** (7 bit)	R/W
i2cadr.0	gc	**General Call Address Acknowledge** If this bit is set, the general call address is recognized; otherwise it is ignored.	R/W

2.4.19 I2CCON - I2C Control Register

Address – F4h
Reset value – 00h
optional register

The I2CCON register controls the operation of I2C interface.

The CPU can read from and write to this 8-bit, directly addressable SFR. Two bits of this register are affected by the I2C hardware: the **si** bit is set when serial interrupt is requested, and the "sto" bit is cleared when STOP condition is present on the I2C bus.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

The I2CCON register is allocated in the SFR memory space when I2C is selected during IP core configuration.

Table 14. I2CCON Register

Bit	Symbol	Description	Type
i2ccon.7	cr2	**Clock rate bit 2**	R/W
i2ccon.6	ens1	**I2C enable bit** When ens1='0' the "sdao" and "sclo" outputs are set to 1, that drives the output pads of the chip in high impedance, and "sdai" and "scli" input signals are ignored. When ens1='1' I2C component is enabled.	R/W
i2ccon.5	sta	**START Flag** When sta='1', the I2C component checks the I2C bus status and if the bus is free a START condition is generated.	R/W
i2ccon.4	sto	**STOP Flag** When sto='1' and I2C interface is in master mode, a STOP condition is transmitted to the I2C bus.	R/W
i2ccon.3	si	**Serial Interrupt Flag** The "si "is set by hardware when one of 25 out of 26 possible I2C states is entered (2.4.17). The only state that does not set the "si" is state F8h, which indicates that no relevant state information is available. The "si" flag must be cleared by software. in order to clear the "si" bit, '0' must be written to this bit. Writing a '1' to si bit does not change value of the "si".	R/W
i2ccon.2	aa	**Assert Acknowledge Flag** When aa='1', an "acknowledge" will be returned when: - the "own slave address" has been received - the general call address has been received while gc bit in i2caddr register was set - a data byte has been received while I2C was in master receiver mode - a data byte has been received while I2C was in slave receiver mode When aa='0', an "not acknowledge" will be returned when: - a data byte has been received while I2C was in master receiver mode - a data byte has been received while I2C was in slave receiver mode	R/W
i2ccon.1	cr1	**Clock rate bit 1**	R/W
i2ccon.0	cr0	**Clock rate bit 0**	R/W

For details on clock rate settings see section 5.17.4 g).

2.4.20 I2CDAT - I2C Data Register

Address – F2h
Reset value – 00h

optional register

The I2CDAT register contains a byte to be transmitted through I2C bus or a byte which has just been received through I2C bus. The CPU can read from and write to this 8-bit, directly addressable SFR while it is not in the process of byte shifting. The I2CDAT register is not shadowed or double buffered so the user should only read I2CDAT when an I2C interrupt occurs.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

I2CDAT register is allocated in the SFR memory space when I2C is selected during IP core configuration.

2.4.21 I2CSTA - I2C Status Register

Address – F5h

Reset value – F8h

optional register

The contents of this register reflect the actual state of I2C interface.

Detailed description of I2C interface functionality can be found in I2C subcomponent specification - 5.14.

The I2CSTA register is allocated in the SFR memory space when I2C is selected during IP core configuration.

Table 15. I2CSTA Register

Bit	Symbol	Description	Type
i2csta.7 i2csta.6 i2csta.5 i2csta.4 i2csta.3	-	**I2C Status Code**	R
i2csta.2 i2csta.1 i2csta.0	-	Not implemented, read as 0	R

2.4.22 IEN0 - Interrupt Enable 0 Register

Address – A8h

Reset value – 00h

The presence of external interrupt enable flags depends on IP core configuration (the number of external interrupts).

The Timer 0 interrupt enable flag is implemented when Timer 0 is selected during IP core configuration.

The Timer 1 interrupt enable flag is implemented when Timer 1 is selected during IP core configuration.

The Timer 2 interrupt enable flag is implemented when Timer 2 is selected during IP core configuration.

The Serial Port 0 interrupt enable flag is implemented when Serial Port 0 is selected during IP core configuration.

The PCA interrupt enable flag is implemented when PCA is selected during IP core configuration.

Table 16. IEN0 Register

Bit	Symbol	Description	Type
ien0.7	eal	**Interrupts enable** When set to 0 – all interrupts are disabled Otherwise enabling each interrupt is done by setting the corresponding interrupt enable bit	R/W
ien0.6	ec	**PCA interrupt enable** When ec=0 PCA interrupt is disabled. When ec=1 and eal=1 PCA interrupt is enabled.	R/W
ien0.5	et2	**TIMER2 interrupt enable** When et2=0 timer2 interrupt is disabled. When et2=1 and eal=1 timer2 interrupt is enabled.	R/W
ien0.4	es0	**Serial Port 0 interrupt enable** When es0=0 Serial Port 0 interrupt is disabled. When es0=1 and eal=1 Serial Port 0 interrupt is enabled.	R/W
ien0.3	et1	**TIMER1 overflow interrupt enable** When et1=0 timer0 overflow interrupt is disabled. When et1=1 and eal=1 timer1 overflow interrupt is enabled.	R/W
ien0.2	ex1	**External interrupt 1 enable** When ex1=0 external interrupt 1 is disabled. When ex1=1 and eal=1 external interrupt 1 is enabled.	R/W
ien0.1	et0	**TIMER0 overflow interrupt enable** When et0=0 timer0 overflow interrupt is disabled. When et0=1 and eal=1 timer0 overflow interrupt is enabled.	R/W
ien0.0	ex0	**External interrupt 0 enable** When ex0=0 external interrupt 0 is disabled. When ex0=1 and eal=1 external interrupt 0 is enabled.	R/W

2.4.23 IEN1 - Interrupt Enable 1 Register

Address – E8h
Reset value – 00h
optional register

IEN1 register is allocated in the SFR memory space when number of external interrupts is more than 2 or RTC is selected during IP core configuration.

The presence of external interrupt enable flags depends on IP core configuration (the number of external interrupts).

The RTC interrupt enable flag is implemented when RTC is selected during IP core configuration.

Table 17. IEN1 Register

Bit	Symbol	Description	Type
ien1.7	-	not used, read as 0	R/W
ien1.6	ertc	**RTC interrupt enable** When ertc=0 RTC interrupt is disabled. When ertc=1 and eal=1 RTC interrupt is enabled.	R/W
ien1.5	ex6	**External interrupt 6 enable** When ex6=0 external interrupt 6 is disabled. When ex6=1 and eal=1 external interrupt 6 is enabled.	R/W
ien1.4	ex5	**External interrupt 5 enable** When ex5=0 external interrupt 5 is disabled. When ex5=1 and eal=1 external interrupt 5 is enabled.	R/W
ien1.3	ex4	**External interrupt 4 enable** When ex4=0 external interrupt 4 is disabled. When ex4=1 and eal=1 external interrupt 4 is enabled.	R/W
ien1.2	ex3	**External interrupt 3 enable** When ex3=0 external interrupt 3 is disabled. When ex3=1 and eal=1 external interrupt 3 is enabled.	R/W
ien1.1	ex2	**External interrupt 2 enable** When ex2=0 external interrupt 2 is disabled. When ex2=1 and eal=1 external interrupt 2 is enabled.	R/W
ien1.0	ex7	**External interrupt 7 enable** When ex7=0 external interrupt 7 is disabled. When ex7=1 and eal=1 external interrupt 7 is enabled.	R/W

2.4.24 IEN2 - Interrupt Enable 2 Register

Address – 9Ah
Reset value – 00h
optional register

IEN2 register is allocated in the SFR memory space when number of external interrupts is more than 8 or Serial 1 is selected during IP core configuration.

The presence of external interrupt enable flags depends on IP core configuration (the number of external interrupts).

The Serial Port 1 interrupt enable flag is implemented when Serial Port 1 is selected during IP core configuration.

Table 18. IEN2 Register

Bit	Symbol	Description	Type
ien2.7	-	not used, read as 0	R
ien2.6	ex13	**External interrupt 13 enable** When ex13=0 external interrupt 13 is disabled. When ex13=1 and eal=1 external interrupt 13 is enabled.	R/W

Bit	Symbol	Description	Type
ien2.5	ex12	**External interrupt 12 enable** When ex12=0 external interrupt 12 is disabled. When ex12=1 and eal=1 external interrupt 12 is enabled.	R/W
ien2.4	ex11	**External interrupt 11 enable** When ex11=0 external interrupt 11 is disabled. When ex11=1 and eal=1 external interrupt 11 is enabled.	R/W
ien2.3	ex10	**External interrupt 10 enable** When ex10=0 external interrupt 10 is disabled. When ex10=1 and eal=1 external interrupt 10 is enabled.	R/W
ien2.2	ex9	**External interrupt 9 enable** When ex9=0 external interrupt 9 is disabled. When ex9=1 and eal=1 external interrupt 9 is enabled.	R/W
ien2.1	ex8	**External interrupt 8 enable** When ex8=0 external interrupt 8 is disabled. When ex8=1 and eal=1 external interrupt 8 is enabled.	R/W
ien2.0	es1	**Serial Port 1 interrupt enable** When es1=0 Serial Port 1 interrupt is disabled. When es1=1 and eal=1 Serial Port 1 interrupt is enabled.	R/W

2.4.25 IPH0, IPL0 - Interrupt Priority Registers

IPH0:

Address – B7h
Reset value – 00h

IPL0:

Address – B8h
Reset value – 00h

The 21 interrupt sources are grouped into 7 priority groups. For each of the groups, one of four priority levels can be selected. It is achieved by setting appropriate values in IPH0 and IPL0 registers.

The contents of the Interrupt Priority Registers define the priority levels for each interrupt source according to the tables below.

Table 19. IPH0 Register

Bit	Symbol	Description	Type
iph0.7	-	not used, read as 0	R
iph0.6	-	**Interrupt priority** Each bit together with corresponding bit from IPL0 register specifies the priority level of the respective interrupt priority group.	R/W
iph0.5	-		
iph0.4	-		
iph0.3	-		
iph0.2	-		
iph0.1	-		
iph0.0	-		

Table 20. IPL0 Register

Bit	Symbol	Description	Type
ipl0.7	-	not used, read as 0	R
ipl0.6	-	**Interrupt priority** Each bit together with corresponding bit from IPH0 register specifies the priority level of the respective interrupt priority group.	R/W
ipl0.5	-		
ipl0.4	-		
ipl0.3	-		
ipl0.2	-		
ipl0.1	-		
ipl0.0	-		

Table 21. Priority Groups

Group	Corresponding interrupt bits			
0	iph0.0, ipl0.0	External interrupt 0	Serial 1 interrupt	External interrupt 7
1	iph0.1, ipl0.1	Timer 0 interrupt	External interrupt 8	External interrupt 2
2	iph0.2, ipl0.2	External interrupt 1	External interrupt 9	External interrupt 3
3	iph0.3, ipl0.3	Timer 1 interrupt	External interrupt 10	External interrupt 4
4	iph0.4, ipl0.4	Serial 0 interrupt	External interrupt 11	External interrupt 5
5	iph0.5, ipl0.5	Timer 2 interrupt	External interrupt 12	External interrupt 6
6	iph0.6, ipl0.6	PCA interrupt	External interrupt 13	RTC interrupt

Table 22. Priority Levels

iph0.x	ipl0.x	
0	0	Level 0 (lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest)

x is the number of priority group.

2.4.26 IRCON - Interrupt Request Control Register

Address – C0h
Reset value – 00h
optional register

IRCON register is allocated in the SFR memory space when number of external interrupts is more than 2 during IP core configuration.

The presence of external interrupt edge flags depends on IP core configuration (the number of external interrupts).

Table 23. IRCON Register

Bit	Symbol	Description	Type
ircon.7	i3fr	**Active edge selection** for external interrupt "int3" 0 - falling edge 1 - rising edge	R/W
ircon.6	i2fr	**Active edge selection** for external interrupt "int2" 0 - falling edge 1 - rising edge	R/W
ircon.5	iex6	External interrupt 6 edge flag	R/W
ircon.4	iex5	External interrupt 5 edge flag	R/W
ircon.3	iex4	External interrupt 4 edge flag	R/W
ircon.2	iex3	External interrupt 3 edge flag	R/W
ircon.1	iex2	External interrupt 2 edge flag	R/W
ircon.0	iex7	External interrupt 7 edge flag	R/W

2.4.27 IRCON2 - Interrupt Request Control 2 Register

Address – BFh
Reset value – 00h
optional register

IRCON2 register is allocated in the SFR memory space when number of external interrupts is more than 8 during IP core configuration.

The presence of external interrupt edge flags depends on IP core configuration (the number of external interrupts).

Table 24. IRCON2 Register

Bit	Symbol	Description	Type
ircon2.7 ircon2.6	-	Not implemented, read as 0	R
ircon2.5	iex13	External interrupt 13 edge flag	R/W
ircon2.4	iex12	External interrupt 12 edge flag	R/W
ircon2.3	iex11	External interrupt 11 edge flag	R/W
ircon2.2	iex10	External interrupt 10 edge flag	R/W
ircon2.1	iex9	External interrupt 9 edge flag	R/W
ircon2.0	iex8	External interrupt 8 edge flag	R/W

2.4.28 P0, P1, P2, P3 – Port Registers

P3:

Address – B0h
Reset value – FFh
optional register

P2:

Address – A0h
Reset value – FFh
optional register

P1:

Address – 90h
Reset value – FFh
optional register

P0:

Address – 80h
Reset value – FFh
optional register

After write operation, the contents of these registers can be observed at the corresponding pins of the chip (port 0, port 1, port2, port3). Writing a '1' to any of the port bits causes the corresponding pin to be at high level, and writing a '0' causes the corresponding pin to be held at low level.

In case of reading, the state of P0, P1, P2, P3 registers reflects the value of the corresponding R80251XC port.

It should be remembered that some of R80251XC instructions (the Read-Modify-Write instructions) while referring to the Port N in fact refer to the Port N register (e.g. INC P0; ANL P2, A) while the others refer directly to the external port input (e.g. MOV A, P1).

The P0, P1, P2 and P3 registers are allocated in the SFR memory space when Port 0, Port 1, Port 2 or Port3 respectively are selected during IP core configuration.

2.4.29 PCON - Power Control Register

Address – 87h
Reset value – 08h

Table 25. PCON Register

Bit	Symbol	Description	Type
pcon.7	s0mod1	**Serial Port 0 baud rate select** (Table 44) (baud rate doubler)	R/W
pcon.6	s0mod0	**Serial Port 0 bit select** When set, the accesses to the S0CON.7 bit applay to the FE bit. When clear, the accesses to the S0CON.7 bit applay to the SM0 bit.	R/W
pcon.5	isr_tm	**Interrupt Service Routine Test Mode flag** When set to 1, the interrupt vectors assigned to Timer 0, 1 & 2, PCA, Serial Port 0 & 1, SPI and I2C interfaces can be triggered only with the use of external inputs of the core.	R/W
pcon.4	-	-	-
pcon.3 pcon.2	gf1 gf0	**General Purpose Flags**	R/W
pcon.1	stop	**Stop mode control** Setting this bit activates the Stop Mode. This bit is always read as 0	R/W

Bit	Symbol	Description	Type
pcon.0	idle	**Idle mode control** Setting this bit activates the Idle Mode. This bit is always read as 0	R/W

2.4.30 PSW, PSW1 - Program Status Word Registers

PSW1:

Address	**–**	**D1h**
Reset value	**–**	**00h**

PSW:

Address	**–**	**D0h**
Reset value	**–**	**00h**

The PSW and the PSW1 registers contain status bits that reflect the current state of the CPU. Note that the Parity bit can only be modified by hardware upon the state of ACC register

The Program Status Word (PSW) register and the Program Status Word 1 (PSW1) register contain four types of bits:

- CY, AC, OV, N, and Z are flags set by hardware to indicate the result of an operation.
- The P bit indicates the parity of the accumulator.
- Bits RS0 and RS1 are programmed by software to select the active register bank for registers R0–R7.
- F0 and UD are available to the user as general-purpose flags.

Individual bits can be addressed with the bit instructions. The PSW and PSW1 bits are used implicitly in the conditional jump instructions.

The PSW register is identical to the PSW register in MCS 51 microcontrollers. The PSW1 register exists only in MCS 251 microcontrollers. Bits CY, AC, RS0, RS1, and OV in PSW1 are identical to the corresponding bits in PSW; i.e., the same bit can be accessed in either register. Table 5-10 lists the instructions that affect the CY, AC, OV, N, and Z bits.

Table 26. PSW Register

Bit	Symbol	Description	Type
psw.7	cy	**Carry flag** Carry bit in arithmetic operations and accumulator for Boolean operations. The carry flag is set by an addition instruction (ADD, ADDC) if there is a carry out of the MSB. It is set by a subtraction (SUB, SUBB) or compare (CMP) if a borrow is needed from the MSB. The carry flag is also affected by some rotate and shift instructions, logical bit instructions, bit move instructions, and the multiply (MUL) and decimal adjust (DA) instructions.	R/W
psw.6	ac	**Auxiliary Carry flag** Set if there is a carry-out from 3rd bit of Accumulator in BCD operations The auxiliary carry flag is affected only by instructions that address 8-bit operands. The AC flag is set if an arithmetic instruction with an 8-bit operand produces a carry out of bit 3 (from addition) or a borrow into bit 3 (from subtraction). Otherwise, it is cleared.	R/W

Bit	Symbol	Description	Type
psw.5	f0	**General purpose Flag 0** General purpose flag available for user	R/W
psw.4	rs1	**Register bank select** control bit 1, used to select working register bank	R/W
psw.3	rs0	**Register bank select** control bit 0, used to select working register bank	R/W
psw.2	ov	**Overflow flag** This bit is set if an addition or subtraction of signed variables results in an overflow error. The overflow flag is also set if a multiplication product overflows one byte or if a division by zero is attempted.	R/W
psw.1	ud	**User-definable Flag:** General purpose flag available for user.	R/W
psw.0	p	**Parity flag** Reflects the number of '1's in the Accumulator. P = '1' if Accumulator contains an odd number of '1's P = '0' if Accumulator contains an even number of '1's Not all instructions update the parity bit. The parity bit is set or cleared by instructions that change the contents of the accumulator (ACC, Register R11).	R

Table 27. PSW1 Register

Bit	Symbol	Description	Type
psw1.7	cy	**Carry flag** Exactly the same as PSW	R/W
psw1.6	ac	**Auxiliary Carry flag** Exactly the same as PSW	R/W
psw1.5	n	**Negative Flag:** This bit is set if the result of the last logical or arithmetic operation was negative (i.e., bit 15 = 1). Otherwise it is cleared.	R/W
psw1.4	rs1	**Register bank select** Exactly the same as PSW	R/W
psw1.3	rs0	**Register bank select** Exactly the same as PSW	R/W
psw1.2	ov	**Overflow flag** Exactly the same as PSW	R/W
psw1.1	z	**Zero Flag:** This flag is set if the result of the last logical or arithmetic operation is zero. Otherwise it is cleared.	R/W
psw1.0	-	-	-

Table 28. Effects of Instructions on PSW and PSW1 Flags

Instruction Type	Instruction	Flags Affected (1), (5)				
		CY	OV	AC (2)	N	Z
Arithmetic	ADD, ADDC, SUB, SUBB, CMP	X	X	X	X	X

Instruction	Instruction	Flags Affected (1), (5)				
	INC, DEC				X	X
	MUL, DIV (3)	0	X		X	X
	DA	X			X	X
Logical	ANL, ORL, XRL, CLR A, CPL A, RL, RR, SWAP				X	X
	RLC, RRC, SRL, SLL, SRA (4)	X			X	X
Program Control	CJNE	X			X	X
	DJNE				X	X

NOTES:

1. X = the flag can be affected by the instruction.
   0 = the flag is cleared by the instruction.
2. The AC flag is affected only by operations on 8-bit operands.
3. If the divisor is zero, the OV flag is set and the other bits are meaningless.
4. For SRL, SLL, and SRA instructions, the last bit shifted out is stored in the CY bit.
5. The parity bit (PSW.0) is set or cleared by instructions that change the contents of the accumulator (ACC - Register R11).

The state of rs1 and rs0 bits selects the working register bank as follows:

Table 29. Register Bank Locations

rs1	rs0	Selected Register Bank	Location
0	0	Bank 0	(00H – 07H)
0	1	Bank 1	(08H – 0FH)
1	0	Bank 2	(10H – 17H)
1	1	Bank 3	(18H – 1FH)

2.4.31 RCAP2H, RCAP2L – Timer 2 (251) reload/capture registers

RCAP2H:

Address – CBh

Reset value – 00h

optional register

CL:

Address – CAh

Reset value – 00h

optional register

The RCAP2H and RCAP2L registers are allocated in the SFR memory space when Timer 2 (251) respectively is selected during IP core configuration.

2.4.32 RTCDATA - RTC Data Register

Address – CFh

Reset value – 00h

optional register

The RTCDAT register is implemented when the Real Time Clock is selected during IP core configuration.

The purpose of the RTCDAT register is to provide (together with the RTCSEK register) a port to read/write the internal registers of the RTC module.

a) RTAH - Real Time Alarm Hour Register

Address – CFh
Reset value – 00h
RTCSEL – 3h
optional register

Table 30. RTAH Register

Bit	Symbol	Description	Type
rtah.7 rtah.6 rtah.5	-	Not used, read as 0	R
rtah.4 rtah.3 rtah.2 rtah.1 rtah.0	-	These bits represent an hour value of alarm that will be compared against corresponding bits of the Real Time Clock Hour Register (rtch.4-0) provided that hce bit (rtcc.4) is set. This register is reset only by the "rtcreset".	R/W

b) RTAM - Real Time Alarm Minute Register

Address – CFh
Reset value – 00h
RTCSEL – 2h
optional register

Table 31. RTAM Register

Bit	Symbol	Description	Type
rtam.7 rtam.6	-	Not used, read as 0	R
rtam.5 rtam.4 rtam.3 rtam.2 rtam.1 rtam.0	-	These bits represent a minute value of alarm that will be compared against corresponding bits of the Real Time Clock Alarm Register (rtcm.5-0) provided that mce bit (rtcc.5) is set. This register is reset only by the "rtcreset".	R/W

c) RTAS - Real Time Alarm Second Register

Address – CFh
Reset value – 00h
RTCSEL – 1h
optional register

Table 32. RTAS Register

Bit	Symbol	Description	Type
rtas.7 rtas.6	-	Not used, read as 0	R
rtas.5 rtas.4 rtas.3 rtas.2 rtas.1 rtas.0	-	These bits represent a second value of alarm that will be compared against corresponding bits of the Real Time Clock Second Register (rtcs.5-0) provided that sce bit (rtcc.6) is set. This register is reset only by the "rtcreset".	R/W

d) RTASS - Real Time Alarm Subsecond Register

Address – CFh
Reset value – 00h
RTCSEL – 0h
optional register

Table 33. RTASS Register

Bit	Symbol	Description	Type
rtass.7 rtass.6 rtass.5 rtass.4 rtass.3 rtass.2 rtass.1 rtass.0	-	These bits represent a subsecond value of alarm that will be compared against corresponding bits of the Real Time Clock Subsecond Register (rtcss.7-0) provided that ssce bit (rtcc.7) is set. This register is reset only by the "rtcreset".	R/W

e) RTCC - Real Time Clock Control Register

Address – CFh
Reset value – 00h
RTCSEL – 4h
optional register

Table 34. RTCC Register

Bit	Symbol	Description	Type
rtcc.7	ssce	**Subsecond Compare Enable** When set to 1 enables a match between rtcss and rtass registers when evaluating an alarm condition; when set to 0 disables this match and thus an interrupt is generated at every subsecond tick as long as other registers match. This bit is reset only by the "rtcreset".	R/W

Bit	Symbol	Description	Type
rtcc.6	sce	**Second Compare Enable** When set to 1 enables a match between rtcs and rtas registers when evaluating an alarm condition; when set to 0 disables this match and thus an interrupt per second is generated as long as other registers match. This bit is reset only by the "rtcreset".	R/W
rtcc.5	mce	**Minute Compare Enable** When set to 1 enables a match between rtcm and rtam registers when evaluating an alarm condition; when set to 0 disables this match and thus one interrupt per minute is generated as long as other registers match. This bit is reset only by the "rtcreset".	R/W
rtcc.4	hce	**Hour Compare Enable** When set to 1 enables a match between rtch and rtah registers when evaluating an alarm condition; when set to 0 disables this match and thus one interrupt per hour is generated as long as other registers match. This bit is reset only by the "rtcreset".	R/W
rtcc.3	rtcre	**Real Time Clock Read Enable** When this bit is set RTC halts updating time registers so that they can be written to. Reading should be accomplished within 1 ms after setting the bit. The bit should be reset after reads are accomplished although it will be automatically reset after 1.95 ms.Reads from RTC registers (rtcss, rtcs, rtcm, rtch, rtcd0, rtcd1) when the rtcre bit is 0 are prohibited and may return erroneous values. This bit is reset also by the "rtcreset".	R/W
rtcc.2	rtcwe	**Real Time Clock Write Enable** When this bit is set RTC halts updating time registers (rtcss, rtcs, rtcm, rtch, rtcd0, rtcd1) so they can be updated. Updates should be accomplished within 1 ms after setting the bit. The bit should be reset after desired updates are completed although it will be automatically reset after 1.95 ms. Resetting this bit will reset the rtcss register to 00H.When the bit value is 0, writes to the RTC register will be ignored. Attempts to set rtcwe and rtcre bits simultaneously will be ignored. This bit is reset also by the "rtcreset".	R/W
rtcc.1	rtcif	**Real Time Clock Interrupt Flag** The bit is set when a match has been detected between all enabled alarm registers and their corresponding clock registers. It must be cleared by software following an interrupt. Clearing all compare enable bits (rtcc.7-4) will also clear this bit. RTC interrupts cannot be generated by setting this bit (as it cannot be set in software). This bit is reset only by the "rtcreset".	R/W
rtcc.0	rtce	**Real Time Clock Enable** This bit enables (or disables) RTC operation as it works as an enable signal for all registers triggered by the clock signal (rtcx) from the real time clock oscillator. This bit is reset only by the "rtcreset".	R/W

f) RTCD0, RTCD1 - Real Time Clock Day Registers 0 & 1

RTCD0:

Address – CFh
Reset value – 00h
RTCSEL = Ah
optional register

RTCD1:

Address – CFh
Reset value – 00h
RTCSEL = Bh
optional register

Table 35. RTCD0 Register

Bit	Symbol	Description	Type
rtcd0.7 rtcd0.6 rtcd0.5 rtcd0.4 rtcd0.3 rtcd0.2 rtcd0.1 rtcd0.0	-	The rtcd0 register bits represents the less significant part of the 16-bit current day count. The day count should be seen as a value set by the user relative to a user specified calendar date. It is up to user software to interpret its contents in terms of an absolute date. No alarm register is compared against these bits.	R/W

Table 36. RTCD1 Register

Bit	Symbol	Description	Type
rtcd1.7 rtcd1.6 rtcd1.5 rtcd1.4 rtcd1.3 rtcd1.2 rtcd1.1 rtcd1.0	-	The rtcd1 register bits represents the most significant part of the 16-bit current day count. The day count should be seen as a value set by the user relative to a user specified calendar date. It is up to user software to interpret its contents in terms of an absolute date. No alarm register is compared against these bits.	R/W

g) RTCH - Real Time Clock Hour Register

Address – CFh
Reset value – 00h
RTCSEL – 9h
optional register

Table 37. RTCH Register

Bit	Symbol	Description	Type
rtch.7 rtch.6 rtch.5	dow2 dow1 dow0	These bits represent the current day of week and count from 1H to 7H each time the rtch.4-0 rolls over from 17H to 0H. When set to 000B, these bits retain 0 value and the day of week function is disabled. No alarm register is compared against these bits.	R/W
rtch.4 rtch.3 rtch.2 rtch.1 rtch.0	-	These bits represent the hour value of the RTC. This subregister counts from 0H to 17H (23 dec).	R/W

h) RTCM - Real Time Clock Minute Register

Address – CFh
Reset value – 00h
RTCSEL – 8h
optional register

Table 38. RTCM Register

Bit	Symbol	Description	Type
rtcm.7 rtcm.6	-	Not used, read as 0	R
rtcm.5 rtcm.4 rtcm.3 rtcm.2 rtcm.1 rtcm.0	-	These bits represent the minute value of the RTC. The register counts from 0H to 3BH (59 dec). The user is responsible for writing only the values in this range to the rtcm register.	R/W

i) RTCS - Real Time Clock Second Register

Address – CFh
Reset value – 00h
RTCSEL – 7h
optional register

Table 39. RTCS Register

Bit	Symbol	Description	Type
rtcs.7 rtcs.6	-	Not used, read as 0	R
rtcs.5 rtcs.4 rtcs.3 rtcs.2 rtcs.1 rtcs.0	-	These bits represent the second value of the RTC. The register counts from 0H to 3BH (59 dec). The user is responsible for writing only the values in this range to the rtcs register.	R/W

j) RTCSS - Real Time Clock Subsecond Register

Address – CFh
Reset value – 00h
RTCSEL – 6h
optional register

Table 40. RTCSS Register

Bit	Symbol	Description	Type

Bit	Symbol	Description	Type
rtcss.7 rtcss.6 rtcss.5 rtcss.4 rtcss.3 rtcss.2 rtcss.1 rtcss.0	-	These bits represent the subsecond (1/256s) value of the RTC. The register counts from 0H to FFH. It is not possible to write to this register. It is set to 00H whenever the rtcwe bit (rtcc.5) is cleared	R/W

2.4.33 RTCSEL - RTC Select Register

Address – CEh
Reset value – 00h
optional register

The RTCSEL register is implemented when the Real Time Clock is selected during IP core configuration.

The purpose of the RTCSEL register is to provide (together with the RTCDAT register) a port to read/write the internal registers of the RTC module.

Table 41. RTCSEL Register

Bit	Symbol	Description	Type
rtcsel.7 rtcsel.6 rtcsel.5 rtcsel.4	-	Not used, read as 0	R
rtcsel.3 rtcsel.2 rtcsel.1 rtcsel.0	-	A 4-bit pointer to internal registers of the RTC	R/W

The RTCSEL works as a pointer to another internal SFR of the RTC. After writing the requested register's ID to the RTCSEL, it becomes available for reading and writing through the RTCDAT location. The table below lists the internal registers of the RTC module together with their pointer values.

Table 42. Internal RTC Register Locations

Register	ID (RTCSEL value)	Description
RTASS	0h	Real Time Alarm Subsecond register
RTAS	1h	Real Time Alarm Second register
RTAM	2h	Real Time Alarm Minute register
RTAH	3h	Real Time Alarm Hour register
RTCC	4h	Real Time Clock Control register
RTCSS	6h	Real Time Clock Subsecond register
RTCS	7h	Real Time Clock Second register
RTCM	8h	Real Time Clock Minute register

Register	ID (RTCSEL value)	Description
RTCH	9h	Real Time Clock Hour register
RTCD0	Ah	Real Time Clock Day register 0
RTCD1	Bh	Real Time Clock Day register 1

2.4.34 S0ADDR - Serial Port 0 Slave Address Register

Address – A9h
Reset value – 00h
optional register

The S0ADDR register contains slave address for multiprocessor communication.

S0ADDR register is allocated in the SFR memory space when Serial Port 0 is selected during IP core configuration.

2.4.35 S0ADEN - Serial Port 0 Slave Address Mask Register

Address – B9h
Reset value – 00h
optional register

The S0ADEN register contains mask byte of slave address for multiprocessor communication.

S0ADEN register is allocated in the SFR memory space when Serial Port 0 is selected during IP core configuration.

2.4.36 S0BUF - Serial Port 0 Data Buffer

Address – 99h
Reset value – 00h
optional register

Writing data to this register sets data in serial output buffer and starts the transmission through Serial Port 0. Reading from the S0BUF reads data from the serial receive buffer.

S0BUF register is allocated in the SFR memory space when Serial Port 0 is selected during IP core configuration.

2.4.37 S0CON - Serial Port 0 Control Register

Address – 98h
Reset value – 00h
optional register

The S0CON register controls the function of Serial Port 0.

S0CON register is allocated in the SFR memory space when Serial Port 0 is selected during IP core configuration.

Table 43. S0CON Register

Bit	Symbol	Description	Type
s0con.7	sm0	**Serial Port 0 mode select** (see Table 44)	R/W
s0con.6	sm1		
s0con.5	sm20	**Multiprocessor communication enable** (see 5.8.4 c)).	R/W
s0con.4	ren0	**Serial reception enable** If set HIGH serial reception at Serial Port 0 is enabled. Otherwise serial reception at Serial Port 0 is disabled.	R/W
s0con.3	tb80	**Transmitter bit 8** This bit is used while transmitting data through Serial Port 0 in Modes 2 and 3. The state of this bit corresponds with the state of the 9^{th} transmitted bit (e.g. parity check or multiprocessor communication). It is controlled by software.	R/W
s0con.2	rb80	**Received bit 8** This bit is used while receiving data through Serial Port 0 in Modes 2 and 3. It reflects the state of the 9^{th} received bit. In Mode 1, if multiprocessor communication is enabled (sm20 = 0), this bit is the stop bit that was received (5.8.4 c)). In Mode 0 this bit is not used.	R/W
s0con.1	ti0	**Transmit interrupt flag** It indicates completion of a serial transmission at Serial Port 0. It is set by hardware at the end of bit 8 in mode 0 or at the beginning of a stop bit in other modes. It must be cleared by software.	R/W
s0con.0	ri0	**Receive interrupt flag** It is set by hardware after completion of a serial reception at Serial Port 0. It is set by hardware at the end of bit 8 in mode 0 or in the middle of a stop bit in other modes. It must be cleared by software.	R/W

Table 44. Serial Port 0 Modes and Baud Rates

<table>
<tr><th>sm0</th><th>sm1</th><th>Mode</th><th>Description</th><th colspan="2">Baud Rate</th></tr>
<tr><td>0</td><td>0</td><td>Mode 0</td><td>shift register</td><td colspan="2">Fclk/12</td></tr>
<tr><td>0</td><td>1</td><td>Mode 1</td><td>8-bit UART</td><td colspan="2">Variable (details below the table)</td></tr>
<tr><td rowspan="4">1</td><td rowspan="4">0</td><td rowspan="4">Mode 2</td><td rowspan="4">9-bit UART</td><td colspan="2">Depends on s0mod1 (pcon.7) value</td></tr>
<tr><th>s0mod1</th><th>Baud Rate</th></tr>
<tr><td>0</td><td>Fclk/64</td></tr>
<tr><td>1</td><td>Fclk/32</td></tr>
<tr><td>1</td><td>1</td><td>Mode 3</td><td>9-bit UART</td><td colspan="2">Variable (details below the table)</td></tr>
</table>

The baud rate for Serial Port 0 working in modes 1 or mode 3:

$$baud\ rate = \frac{2^{S0MOD1} * Fclk}{32} * (Timer1\ overflow\ rate)$$

2.4.38 S1BUF - Serial Port 1 Data Buffer

Address – 9Ch
Reset value – 00h
optional register

Writing data to this register sets data in serial output buffer and starts the transmission through Serial Port 1. Reading from the S1BUF reads data from the serial receive buffer.

S1BUF register is allocated in the SFR memory space when Serial Port 1 is selected during IP core configuration.

2.4.39 S1CON - Serial Port 1 Control Register

Address – 9Bh
Reset value – 00h
optional register

The S1CON register controls the function of the Serial Port 1.

S1CON register is allocated in the SFR memory space when Serial Port 1 is selected during IP core configuration.

Table 45. S1CON Register

Bit	Symbol	Description	Type
s1con.7	sm	**Serial Port 1 mode select** sm = 0: Mode A selected for Serial Port 1 - 9-bit UART sm = 1: Mode B selected for Serial Port 1 - 8-bit UART	R/W
s1con.6	-	not used, read as 0	R
s1con.5	sm21	**Multiprocessor communication enable** (5.8.4 c)).	R/W
s1con.4	ren1	**Serial reception enable** If set HIGH serial reception at Serial Port 1 is enabled. Otherwise serial reception at Serial Port 1 is disabled.	R/W
s1con.3	tb81	**Transmitter bit 8** This bit is used while transmitting data through Serial Port 1 in Mode A. The state of this bit corresponds with the state of the 9th transmitted bit (e.g. parity check or multiprocessor communication). It is controlled by software.	R/W
s1con.2	rb81	**Received bit 8** This bit is used while receiving data through Serial Port 1 in Mode A. It reflects the state of the 9th received bit. In Mode B, if multiprocessor communication is enabled (sm21 = 0), this bit is the stop bit that was received (5.8.4 c)).	R/W

Bit	Symbol	Description	Type
s1con.1	ti1	**Transmit interrupt flag** It indicates completion of a serial transmission at Serial Port 1. It is set by hardware at the beginning of a stop bit in mode A or B. It must be cleared by software.	R/W
s1con.0	ri1	**Receive interrupt flag** It is set by hardware after completion of a serial reception at Serial Port 1. It is set by hardware in the middle of a stop bit in mode A or B. It must be cleared by software.	R/W

The baud rate for Serial Port 1:

$$baud\ rate = \frac{Fclk}{32 * \left(2^{10} - s1rel\right)}$$

s1rel – the contents of S1REL registers (2.4.40)

2.4.40 S1RELH, S1RELL - Serial Port 1 Reload Register

S1RELL:

Address – 9Dh
Reset value – 00h
optional register

S1RELH:

Address – BBh
Reset value – 03h
optional register

Serial Port Reload Register is used for Serial Port 1 baud rate generation (page 63).

Only 10 bits are used. 8 bits from the S1RELL as lower bits and 2 bits from the S1RELH (s1relh.1, s1relh.0) as higher bits.

S1RELL and S1RELH registers are allocated in the SFR memory space when Serial Port 1 is selected during IP core configuration.

2.4.41 SPH, SP - Stack Pointer Registers

SPH:

Address – BEh
Reset value – 00h

SP:

Address – 81h
Reset value – 07h

Dword register DR60 is the stack pointer, SPX. The byte at location 63 in register file is the 8-bit stack pointer, SP, in the MCS 51 architecture. The byte at location 62 in register file is the stack pointer high, SPH. The two bytes allow the stack to extend to the top of memory region 00:. SP and SPH can be accessed as SFRs.
Two instructions, PUSH and POP directly address the stack pointer. Subroutine calls (ACALL, ECALL, LCALL) and returns (ERET, RET, RETI) also use the stack pointer. to preserve the stack, do not use DR60 as a general-purpose register.

2.4.42 SPCON - Serial Peripheral Control Register

Address – E2h
Reset value – 14h
optional register

The Serial Peripheral Control Register is used to configure the SPI module. It selects the Master clock rate, configures the Module as Master or Slave, selects serial clock polarity and phase, enables the "ssn" input and enables/disables the whole SPI module.

SPCON register is allocated in the SFR memory space when SPI is selected during IP core configuration.

Table 46. SPCON Register

<table>
<tr><th>Bit</th><th>Symbol</th><th>Description</th><th>Type</th></tr>
<tr><td>spcon.7</td><td>spr2</td><td>Serial Peripheral Rate 2
Together with “spr1” and “spr0” defines the clock rate in master mode.</td><td>R/W</td></tr>
<tr><td>spcon.6</td><td>spen</td><td>Serial Peripheral Enable
When cleared disables the SPI interface.
When set enables the SPI interface.</td><td>R/W</td></tr>
<tr><td>spcon.5</td><td>ssdis</td><td>SS Disable
When cleared enables the “ssn” input in both Master and Slave modes. When set disables the “ssn” input in both Master and Slave modes.
In Slave mode, this bit has no effect if “cpha”=0. When “ssdis” is set, no “modf” interrupt request will be generated.</td><td>R/W</td></tr>
<tr><td>spcon.4</td><td>mstr</td><td>Serial Peripheral Master
When cleared configures the SPI as a Slave.
When set configures the SPI as a Master.</td><td>R/W</td></tr>
<tr><td>spcon.3</td><td>cpol</td><td>Clock Polarity
When cleared, the “sck” is set to 0 in idle state.
When set, the “sck” is set to 1 in idle state.</td><td>R/W</td></tr>
<tr><td>spcon.2</td><td>cpha</td><td>Clock Phase
When cleared, data is sampled when the “scki”/”scko” leaves the idle state (see “cpol”). When set, data is sampled when the “scki”/”scko” returns to idle state (see “cpol”).</td><td>R/W</td></tr>
<tr><td>spcon.1
spcon.0</td><td>spr1
spr0</td><td>Serial Peripheral Rate
Together with “spr2” specify the serial clock rate in Master mode.
<table>
<tr><th>spr2</th><th>spr1</th><th>spr0</th><th>Serial Peripheral Rate</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Fclk / 2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Fclk / 4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Fclk / 8</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Fclk / 16</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Fclk / 32</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Fclk / 64</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Fclk / 128</td></tr>
</table></td><td>R/W
R/W</td></tr>
</table>

Bit	Symbol	Description				Type
		1	1	1	the master clock is not generated (when "cpol" = '1' on the "scko" output is high level, otherwise is low level)	

2.4.43 SPDAT - Serial Peripheral Data Register

Address – E3h
Reset value – 00h
optional register

The SPDAT is a read/write buffer for the "receive data" register. While writing to the SPDAT data is placed directly into the shift register (there is no transmit buffer).

Reading the SPDAT returns the value located in the receive buffer, not the shift register.

SPDAT register is allocated in the SFR memory space when SPI is selected during IP core configuration.

2.4.44 SPSSN - Serial Peripheral Slave Select Register

Address – E4h
Reset value – FFh
optional register

The SPSSN is a read/write register used to control the "spssn[7:0]" output bus of the core. Data written to this register is directly available on the "spssn" output. Each of its bits can be used to select a separate external SPI slave device.

The SPSSN register is allocated in the SFR memory space when SPI is selected during IP core configuration.

2.4.45 SPSTA - Serial Peripheral Status Register

Address – E1h
Reset value – 00h
optional register

The SPSTA register contains flags to signal data transfer complete, Write collision and inconsistent logic level on "ssn" (slave select) pin (mode fault error).

SPSTA register is allocated in the SFR memory space when SPI is selected during IP core configuration.

Table 47. SPSTA Register

Bit	Symbol	Description	Type
spsta.7	spif	**Serial Peripheral Data Transfer Flag** Set by hardware upon data transfer completion. Cleared by hardware when data transfer is in progress. Can be also cleared by reading the "spsta" register with the "spif" bit set, and then reading the "spdat" register.	R

Bit	Symbol	Description	Type
spsta.6	wcol	**Write Collision Flag** Set by hardware upon write collision to "spdat". Cleared by hardware upon data transfer completion when no collision has occurred. Can be also cleared by an access to "spsta" register and an access to "spdat" register.	R
spsta.5	sserr	**Synchronous Serial Slave Error Flag** Set by hardware when "ssn" input is deasserted before the end of receive sequence. Cleared by disabling the SPI module (clearing "spen" bit in "spcon" register).	R
spsta.4	modf	**Mode Fault Flag** Set by hardware when the "ssn" pin level is in conflict with actual mode of the SPI_MS controller (configured as master while externally selected as slave). Cleared by hardware when the "ssn" pin is at appropriate level. Can be also cleared by software by reading the "spsta" register with "modf" bit set.	R
spsta.3 spsta.2 spsta.1 spsta.0	-	not used, read as 0	R

2.4.46 SRST - Software Reset Register

Address – F7h
Reset value – 00h
optional register

The software reset will be accomplished through the SRST SFR register. The contents of this register are presented below.

Table 48. SRST Register

Bit	Symbol	Description	Type
srst.7	-	not used, read as 0	R
srst.6	-		
srst.5	-		
srst.4	-		
srst.3	-		
srst.2	-		
srst.1	-		

Bit	Symbol	Description	Type
srst.0	srstreq	Software reset request. Writing '0' value to this bit will have no effect. Single writing '1' value to this bit will have no effect. Double writing '1' value (in two consecutive instructions) will generate an internal software reset. Reading this bit will inform about the reset source: if '0' – source of last reset sequence was not a software reset (hardware, watchdog or debugger reset); if '1' – source of last reset sequence was a software reset (caused by double writing '1' value to the "srstreq" bit).	R/W

2.4.47 T2CON - Timer 2 (515) Control Register

Address – C8h
Reset value – 00h
optional register

T2CON register reflects the current status of R80251XC Timer 2 and it is used to control Timer 2 operation.

T2CON register is allocated in the SFR memory space when Timer 2 is selected during IP core configuration.

Table 49. T2CON Register

Bit	Symbol	Description	Type
t2con.7	t2ps	**Prescaler select** t2ps = 0 – timer 2 is clocked with 1/12 of the oscillator frequency. t2ps = 1 – timer 2 is clocked with 1/24 of the oscillator frequency.	R/W
t2con.6	i3fr	**Active edge selection** for external interrupt "int3", (used also as compare and capture signal) 0 - falling edge 1 - rising edge	R/W
t2con.5	i2fr	**Active edge selection** for external interrupt "int2" 0 - falling edge 1 - rising edge	R/W
t2con.4	t2r1	**Timer 2 reload mode** selection: 0X – reload disabled 10 – Mode 0 11 – Mode 1	R/W
t2con.3	t2r0		
t2con.2	t2cm	**Timer 2 copare mode** selection 0 – Mode 0 1 – Mode 1	R/W
t2con.1	t2i1	**Timer 2 input selection**: (t2i1, t2i0) 00 timer 2 stopped	R/W

Bit	Symbol	Description	Type
t2con.0	t2i0	01 input frequency f/12 or f/24 10 timer 2 is incremented by falling edge detection at pin "t2" 11 input frequency f/12 or f/24 gated by external pin "t2"	

2.4.48 T2MOD - Timer 2 (251) Mode Control Register

Address – C8h
Reset value – 00h
optional register

T2MOD register is allocated in the SFR memory space when Timer 2 (251) is selected during IP core configuration.

2.4.49 TCON – Timers 0 & 1 - Timer/Counter Control Register

Address – 88h
Reset value – 00h
optional register

TCON register reflects the current status of R80251XC Timer 0 and Timer 1 and it is used to control operation of these modules.

TCON register is allocated in the SFR memory space when Timer 0 or Timer 1 is selected during IP core configuration.

Table 50. TCON Register

Bit	Symbol	Description	Type
tcon.7	tf1	**Timer 1 overflow flag** Bit set by hardware when TIMER1 overflows. This flag can be cleared by software and is automatically cleared when interrupt is processed.	R/W
tcon.6	tr1	**TIMER1 Run control** If cleared, Timer 1 stops.	R/W
tcon.5	tf0	**Timer 0 overflow flag** Bit set by hardware when Timer 0 overflows. This flag can be cleared by software and is automatically cleared when interrupt is processed.	R/W
tcon.4	tr0	**Timer 0 Run control** If cleared, Timer 0 stops.	R/W
tcon.3	ie1	**External interrupt 1 flag** Set by hardware, when external interrupt int1 (edge/level, depending on settings) is observed. Cleared by hardware when interrupt is processed.	R/W

Bit	Symbol	Description	Type
tcon.2	it1	**External interrupt 1 type control** If set, external interrupt 1 is activated at falling edge on input pin. If cleared, external interrupt 1 is activated at low level on input pin.	R/W
tcon.1	ie0	**External interrupt 0 flag** Set by hardware, when external interrupt int0 (edge/level, depending on settings) is observed. Cleared by hardware when interrupt is processed.	R/W
tcon.0	it0	**External interrupt 0 type control** If set, external interrupt 0 is activated at falling edge on input pin. If cleared, external interrupt 0 is activated at low level on input pin.	R/W

The tf0, tf1 (timer 0 and timer 1 overflow flags), ie1 and ie1 (external interrupt 0 and 1 flags) will be automatically cleared by hardware when the corresponding service routine is called.

2.4.50 TH0, TL0 – Timer 0 Registers

TH0:

Address	**–**	**8Ch**
Reset value	**–**	**00h**

optional registers

TL0:

Address	**–**	**8Ah**
Reset value	**–**	**00h**

optional registers

These registers reflect the state of Timer 0. TH0 holds higher byte and TL0 holds lower byte.

Timer 0 can be configured to operate as either timer or counter.

TL0 and TH0 registers are allocated in the SFR memory space when Timer 0 is selected during IP core configuration.

2.4.51 TH1, TL1 – Timer 1 Registers

TL1:

Address	**–**	**8Dh**
Reset value	**–**	**00h**

optional register

TH1:

Address	**–**	**8Bh**
Reset value	**–**	**00h**

optional register

These registers reflect the state of Timer 1. TH1 holds higher byte and TL1 holds lower byte.

Timer 1 can be configured to operate as either timer or counter.

TL1 and TH1 registers are allocated in the SFR memory space when Timer 1 is selected during IP core configuration.

2.4.52 TH2, TL2 – Timer 2 Registers

TL2:

Address	**–**	**CCh**
Reset value	**–**	**00h**

optional register

TH2:

Address	**–**	**CDh**
Reset value	**–**	**00h**

optional register

These registers reflect the state of Timer 2. TH2 holds higher byte and TL2 holds lower byte.

Timer 2 can be configured to operate in compare, capture or reload modes.

TL2 and TH2 registers are allocated in the SFR memory space when Timer 2 is selected during IP core configuration.

2.4.53 TMOD – Timers 0 & 1 Mode Register

Address – 89h

Reset value – 00h

optional register

TMOD register is used in configuration of R80251XC Timer 0 and TIMER1.

TMOD register is allocated in the SFR memory space when Timer 0 or Timer 1 is selected during IP core configuration.

Table 51. TMOD Register

Bit	Symbol	Description	Type
tmod.7	gate	**Timer 1 gate control** If set, enables external gate control (pin "int(1)") for Counter 1. When "int(1)" is high, and "tr1" bit is set (Table 50), the Counter 1 is incremented every falling edge on "t1" input pin	R/W
tmod.6	c/t	**Timer 1 counter/timer select** Selects Timer or Counter operation. When set to 1, a Counter operation is performed, when cleared to 0, the Timer/Counter 1 will function as a Timer.	R/W
tmod.5	m1	**Timer 1 mode** Selects mode for Timer/Counter 1, as shown in table below.	R/W
tmod.4	m0		
tmod.3	gate	**Timer 0 gate control** If set, enables external gate control (pin "int(0)") for Counter 0. When "int(0)" is high, and "tr0" bit is set (Table 50), the Counter 0 is incremented every falling edge on "t0" input pin	R/W
tmod.2	c/t	**Timer 0 counter/timer select** Selects Timer or Counter operation. When set to 1, a Counter operation is performed, when cleared to 0, the Timer/Counter 0 will function as a Timer.	R/W
tmod.1	m1	**Timer 0 mode** Selects mode for Timer/Counter 0, as shown in table below.	R/W
tmod.0	m0		

Table 52. Timers/Counters Modes

m0	m1	Mode	Function
0	0	Mode 0	13-bit Counter/Timer, with 5 lower bits in tl0 (tl1) register and 8 bits in th0 (th1) register (for Timer 0 or Timer 1, respectively). Note, that unlike in 80C51, the 3 high-order bits of tl0 (tl1) are zeroed whenever Mode 0 is enabled.
0	1	Mode 1	16-bit Counter/Timer.

m0	m1	Mode	Function
1	0	Mode 2	8 -bit auto-reload Counter/Timer. The reload value is kept in th0 (th1), while tl0 (tl1) is incremented every machine cycle. When tl0 (tl1) overflows, a value from th0 (th1) is copied to tl0 (tl1).
1	1	Mode 3	For TIMER1: TIMER1 is stopped. For TIMER0: Timer 0 acts as two independent 8 bit Timers / Counters – tl0, th0. - tl0 uses the TIMER0 control bits and sets tf0 flag on overflow - th0 operates as timer. It is enabled by tr1 bit and sets tf1 flag on overflow.

2.4.54 UCONFIG0, UCONFIG1 - User Configuration Registers

UCONFIG0:

Address	**–**	**91h**
Reset value	**–**	**FFh**
Memory location	**–**	**FF:FFF8h**

UCONFIG1:

Address	**–**	**92h**
Reset value	**–**	**FFh**
Memory location	**–**	**FF:FFF9h**

Configuration array is fetched after reset from memory at addresses FF:FFF8H to FF:FFFFH. The configuration bytes are located at locations FF:FFF8H and FF:FFF9H.

Table 53 Configuration Byte UCONFIG0

Bit	Symbol	Description	Type
uconfig0.7	ucon	**Configuration Byte Location** This bit is implemented only in the R80251XC-I version (Intel 80C251 timing-compatible). For R80251XC-T version, it is always 0. This bit is written during the first UCONFIG0 read performed after reset. After that, the second UCONFIG0 read and UCONFIG1 is performed: - from on-chip program memory, when 'ucon'=0 or when 'ea' pin=1 regardless of 'ucon' bit previously read. - from off-chip program memory when 'ucon'=1 and 'ea' pin=0.	R/W
uconfig0.6 uconfig0.5	wsa1 wsa0	**Wait State A** (all regions except 01:): For external memory accesses, selects the number of wait states for memrd, memwr. wsa1 | wsa0 | description 0 | 0 | Inserts 3 wait states for all regions except 01: 0 | 1 | Inserts 2 wait states for all regions except 01: 1 | 0 | Inserts 1 wait state for all regions except 01: 1 | 1 | Zero wait states for all regions except 01:	R/W

<table>
<tr><th>Bit</th><th>Symbol</th><th>Description</th><th>Type</th></tr>
<tr><td>uconfig0.4</td><td>xale</td><td>Extend ALE
This bit is implemented only in the R80251XC-I version (Intel 80C251 timing-compatible). For R80251XC-T version, it is always 0.
When ‘xale’=1, the ‘ale’ output pulse takes a single clock cycle.
When ‘xale’=0, the ‘ale’ output pulse takes 3 clock cycles.</td><td>R/W</td></tr>
<tr><td>uconfig0.3
uconfig0.2</td><td>rd[1:0]</td><td>Memory Signal Selection
These bits are implemented only in the R80251XC-I version (Intel 80C251 timing-compatible). For R80251XC-T version, they are always 0.
These bits configure the size of External Memory and behaviour of p1, p3, psen and wr pins
<table>
<tr><th>rd</th><th>p1.7
(a17)</th><th>p3.7
(a16/rd)</th><th>psen</th><th>p3.6
(wr)</th><th>External Memory size</th></tr>
<tr><td>00</td><td>memaddr[17]</td><td>memaddr[16]</td><td rowspan="3">all addresses</td><td rowspan="3">writes to all locations</td><td>256KB</td></tr>
<tr><td>01</td><td></td><td>memaddr[16]</td><td>128KB</td></tr>
<tr><td>10</td><td></td><td></td><td>64KB</td></tr>
<tr><td>11</td><td></td><td>‘rd’ for addresses below 0x7FFFFF</td><td>only addresses above or equal 0x800000</td><td>only writes to MCS51 XDATA locations</td><td>64KB, MCS51 compatible</td></tr>
</table></td><td>R/W</td></tr>
<tr><td>uconfig0.1</td><td>page</td><td>Page Mode Select
This bit is implemented only in the R80251XC-I version (Intel 80C251 timing-compatible). For R80251XC-T version, it is always 0.
If ‘page’=0, the Page Mode is enabled and the address bits 15:8 are multiplexed with data bits 7:0 on port P2, while address bits 7:0 are put to port P0.
If ‘page’=1, the Page Mode is disabled and address bits 15:8 are put to port P2, while address bits 7:0 and data bits 7:0 are multiplexed on port P0.</td><td>R/W</td></tr>
<tr><td>uconfig0.0</td><td>src</td><td>Source Mode/Binary Mode Select:
src = ‘0’ - binary mode (compatible with MCS 51 microcontrollers).
src = ‘1’ - source mode.</td><td>R/W</td></tr>
</table>

Table 54. Configuration Byte UCONFIG1

Bit	Symbol	Description	Type

<table>
<tr><th>Bit</th><th>Symbol</th><th>Description</th><th>Type</th></tr>
<tr><td>uconfig1.7
uconfig1.6
uconfig1.5</td><td></td><td>Not used, read as 0</td><td>R</td></tr>
<tr><td>uconfig1.4</td><td>intr</td><td>Interrupt Mode:
intr = ‘1’ - interrupts push 4 bytes onto the stack (the 3 bytes of the PC and PSW1).
intr = ‘0’ - interrupts push the 2 lower bytes of the PC onto the stack.</td><td>R/W</td></tr>
<tr><td>uconfig1.3</td><td>wsb</td><td>Wait State B
This bit is implemented only in the R80251XC-I version (Intel 80C251 timing-compatible). For R80251XC-T version, it is always 0.
When ‘wsb’=0, one additional external wait state is generated for memory region 01. When ‘wsb’=1, no additional wait states are generated for region 01.</td><td>R/W</td></tr>
<tr><td>uconfig1.2
uconfig1.1</td><td>wsb1
wsb0</td><td>External Wait State B (Region 01:):
For external memory accesses, selects the number of wait states for memrd, memwr.
<table>
<tr><th>wsb1</th><th>wsb0</th><th>description</th></tr>
<tr><td>0</td><td>0</td><td>Inserts 3 wait states for region 01:</td></tr>
<tr><td>0</td><td>1</td><td>Inserts 2 wait states for region 01:</td></tr>
<tr><td>1</td><td>0</td><td>Inserts 1 wait state for region 01:</td></tr>
<tr><td>1</td><td>1</td><td>Zero wait states for region 01:</td></tr>
</table></td><td>R/W</td></tr>
<tr><td>uconfig1.0</td><td></td><td>Not used, read as 1</td><td>R</td></tr>
</table>

2.4.55 WCON – Waitstate Control Register

Address – A7h

Reset value – 00h

2.4.56 WDTRST - Watchdog Timer Reset Register

Address – A6h

Reset value – 00h

optional register

The Watchdog component can be cleared and enabled by two consecutive writes (1Eh, E1h values) to the WDTRST register.

WDTRST register is allocated in the SFR memory space when Watchdog is selected during IP core configuration.

2.4.57 External Special Function Registers

The External Special Function Registers interface services up to 119 off-core special function registers. The off-core peripherals can use all addresses from the SFR address space range 80H to FFH except of those that are already implemented inside the core.

If an SFR is not implemented inside the core, it is routed to External Special Function Registers interface.

When a read instruction occurs with an address of an SFR, which has been implemented both inside and outside the core, the read will return the contents of the internal SFR.

When a write instruction occurs with an address of an SFR which has been implemented both inside and outside the core, both internal and external SFRs will be written.

The External SFR Interface is equipped with a wait-state mechanism, allowing to connect slower peripherals from outside. Each read or write operation on SFR can be completed only when an active "sfrack" is there at the rising edge of the CPU clock ("clkcpu"). Note that the "sfrack" affects all SFRs, internal and external. The external device should set '1' to that input by default, when internal registers are accessed. The "sfrack" signal must be synchronized to the "clkcpu" clock since it is used directly without pre-sampling inside the CPU.

2.5. External Memory Interface (R80251XC-T(F) only)

2.5.1 Interface Description

The R80251XC contains interface to External Memory which is used to access external data memory and program memory.

The interface consists of the 24-bit wide address bus "memaddr", the 8-bit input data bus "memdatai", the 8-bit output data bus "memdatao", and control signals "mempsrd", "mempswr", "memrd", "memwr". This interface should be used to access falling-edge triggered synchronous RAM, or asynchronous RAM if the write strobe is gated by the 'clkcpu' to ensure proper timing..

For rising-edge triggered SRAM, the additional output signals are provided: "memaddr_comb", "memdatao_comb", "mempsrd_comb", "mempswr_comb", "memrd_comb" and "memwr_comb". Data read from memory should be connected to the "memdatai" input.

The External Memory Interface provides hardware and software controlled wait states. The number of software controlled waitstates is defined by the UCONFIG0 and UCONFIG1 registers. Apart from that, there are "mempsack" and "memack" signals used to stretch each memory access by external hardware if necessary. An access can only be finished when the internal (software-controlled) wait state has finished and the corresponding memory acknowledge input is 1 at the rising edge of the "clkcpu".

2.5.2 Program Memory Read Cycle (Negedge SRAM)

The figure below shows an example read of the instruction at address N.

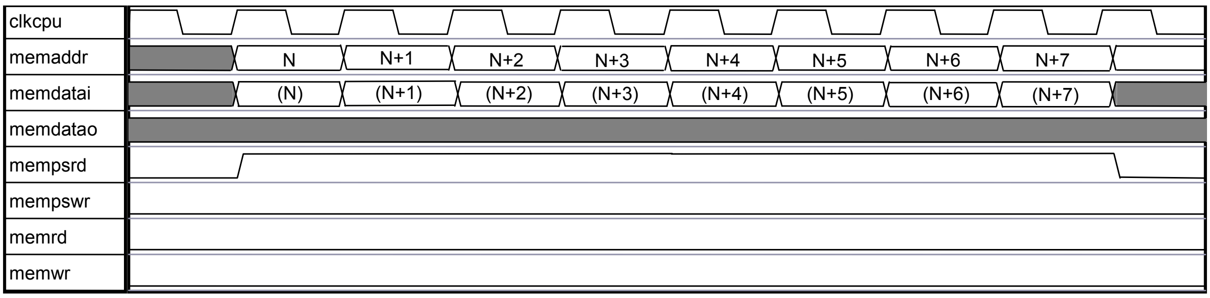


Figure 9. Program Memory Read Cycle (Negedge SRAM)

Note: N - address of actually executed instruction

(N) - instruction fetched from address N

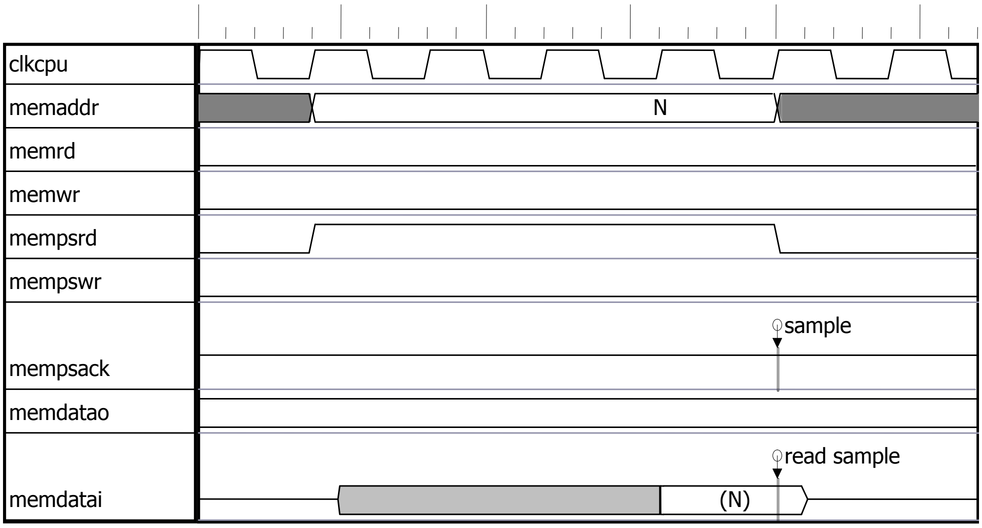


Figure 10. Program Memory Read Cycle with 3 Wait States

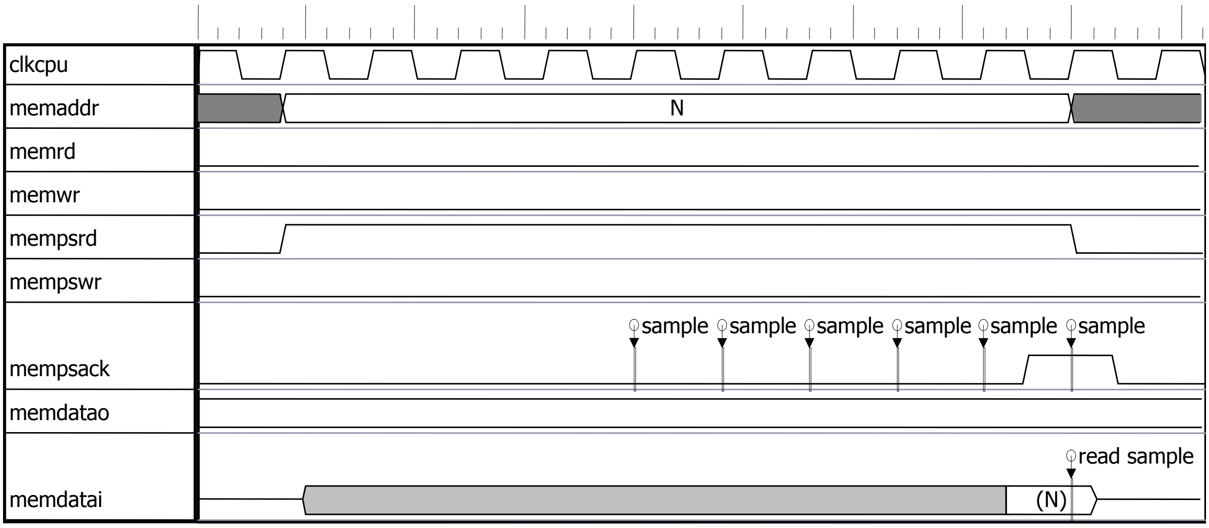


Figure 11. Program Memory Read Cycle with 3 Wait States, Delayed by Mempsack

2.5.3 Program Memory Read Cycle (Posedge SRAM)

The figure below shows an example read of the instruction at address N.

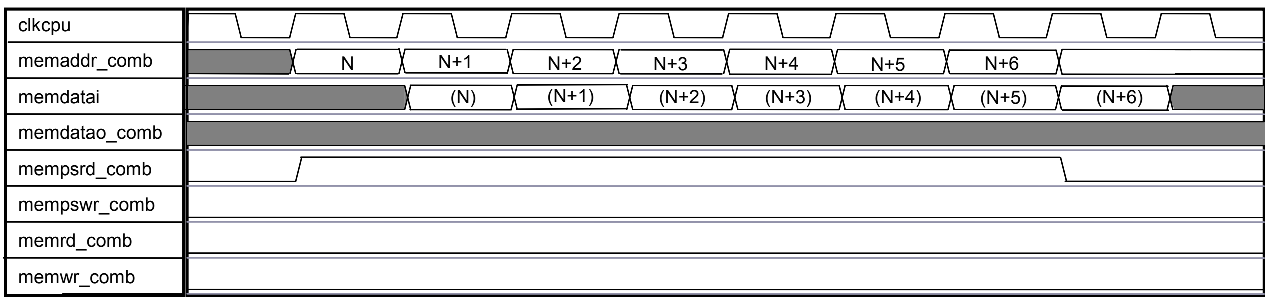


Figure 12. Program Memory Read Cycle (Posedge SRAM)

Note: N - address of actually executed instruction

(N) - instruction fetched from address N

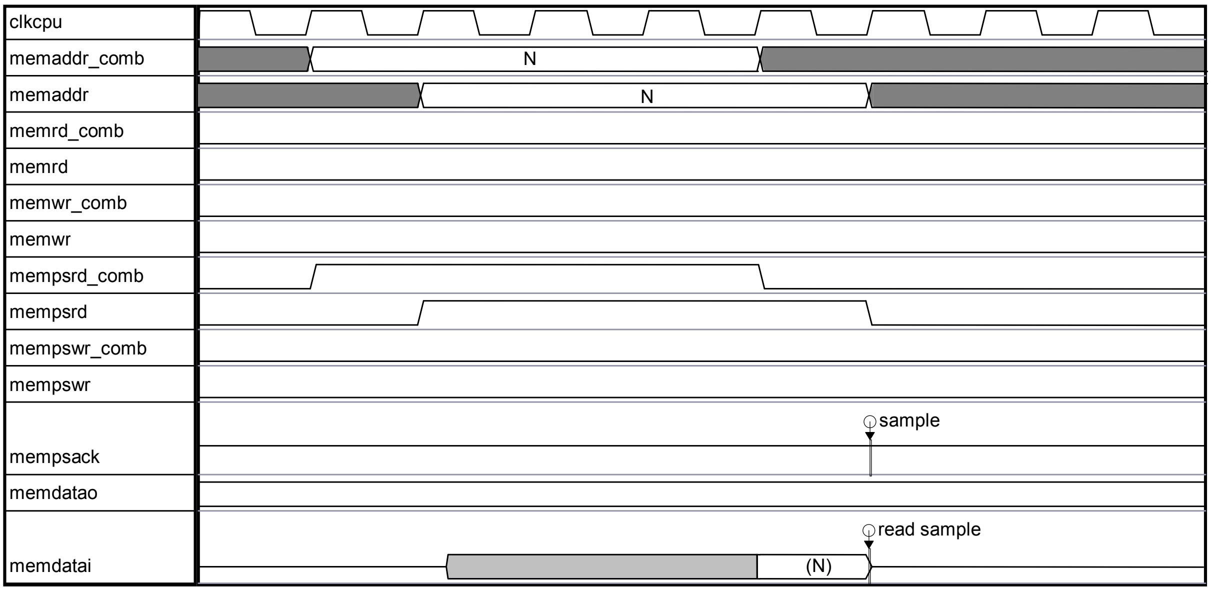


Figure 13. Program Memory Read Cycle with 3 Wait States (Posedge SRAM)

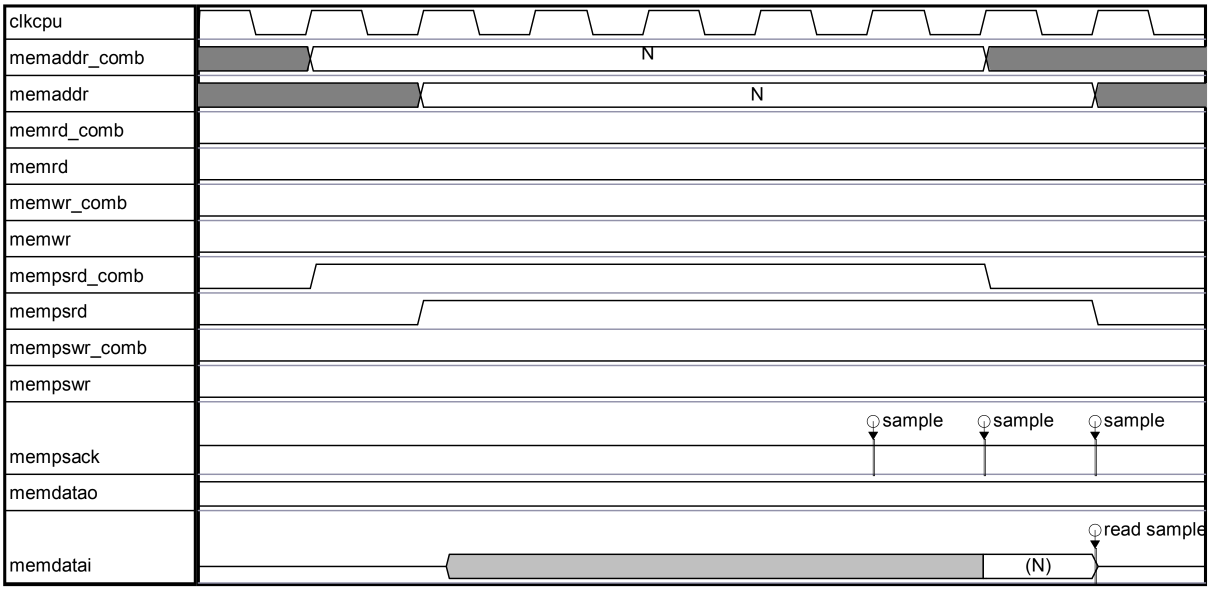


Figure 14. Program Memory Read Cycle with 3 Wait States Delayed by Mempsack(Posedge SRAM)

2.5.4 Program Memory Write Cycle (Negedge SRAM)

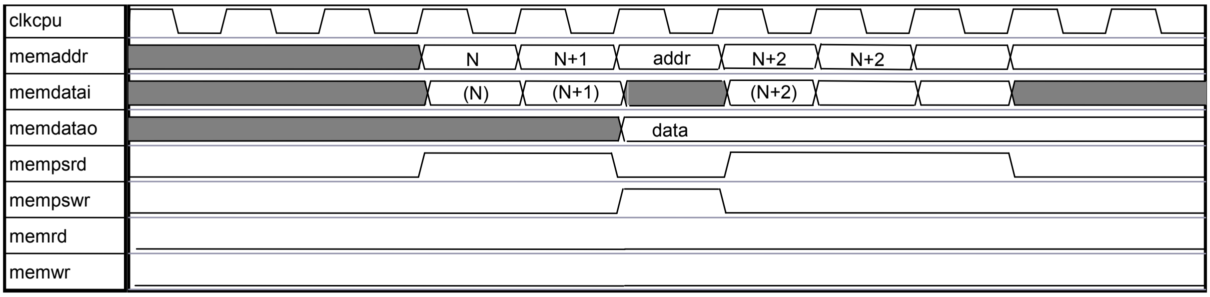


Figure 15. Program Memory Write Cycle (Negedge SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	addr	- address of program memory being written
	Data	- data written at addr

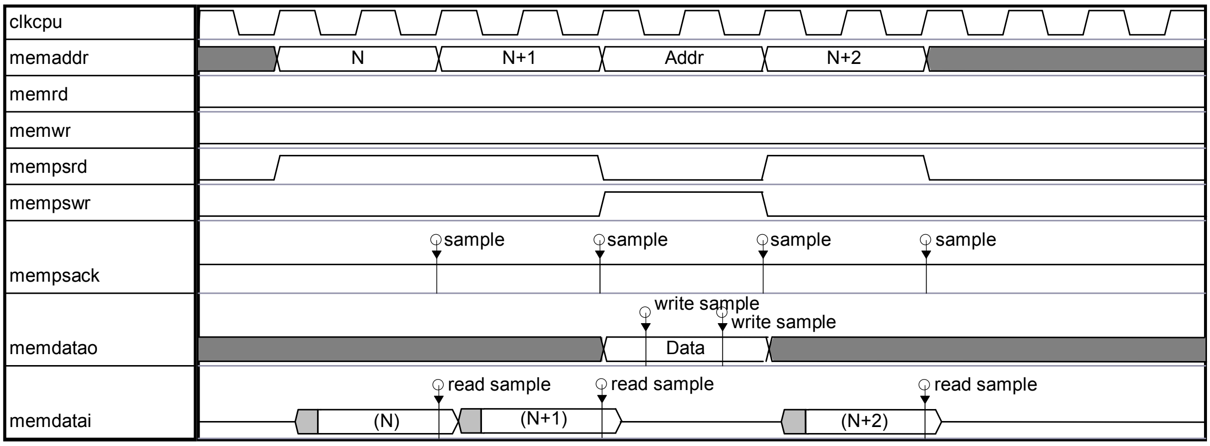


Figure 16. Program Memory Write Cycle with 1 Wait State (Negedge SRAM)

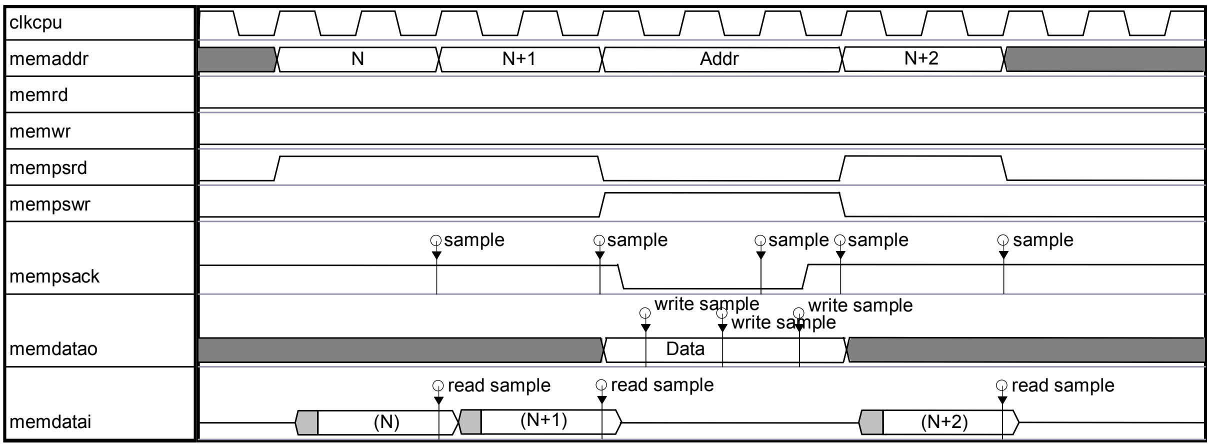


Figure 17. Program Memory Write Cycle with 1 Wait State Delayed by Mempsack (Negedge SRAM)

2.5.5 Program Memory Write Cycle (Posedge SRAM)

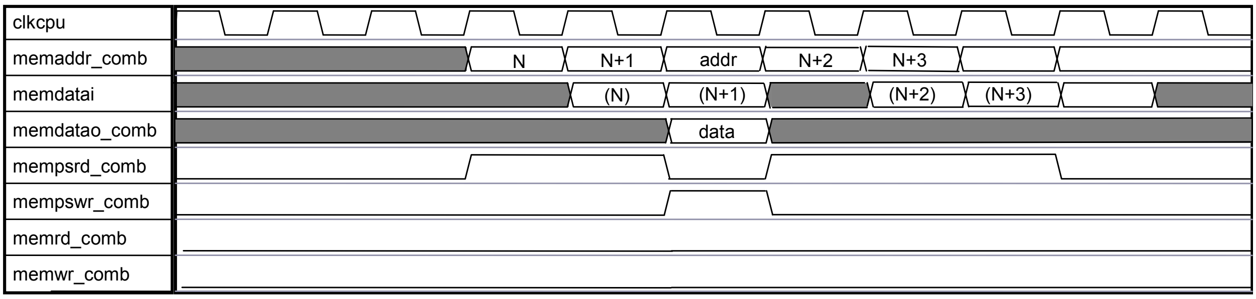


Figure 18. Program Memory Write Cycle (Posedge SRAM)

Note:	N	- address of instruction performing write
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	addr	- address of program memory being written
	Data	- data written at addr

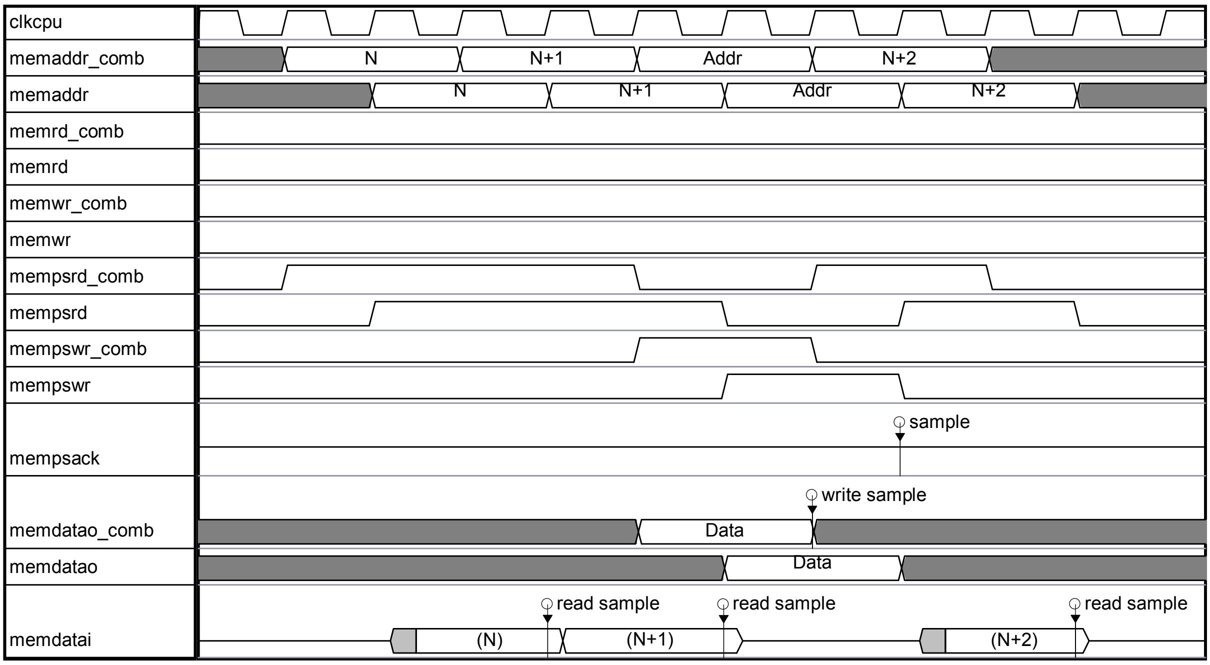


Figure 19. Program Memory Write Cycle with 1 Wait State (Posedge SRAM)

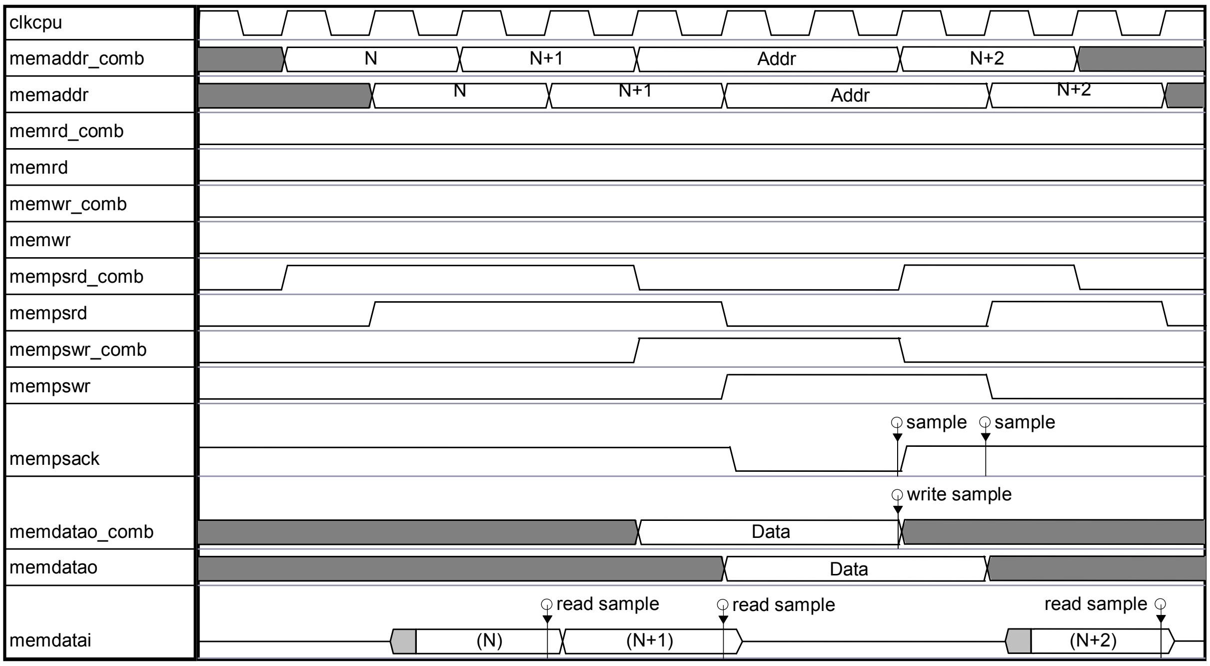


Figure 20. Program Memory Write Cycle with 1 Wait State Delayed by Mempsack (Posedge SRAM)

2.5.6 External Data Memory Read Cycle (Negedge SRAM)

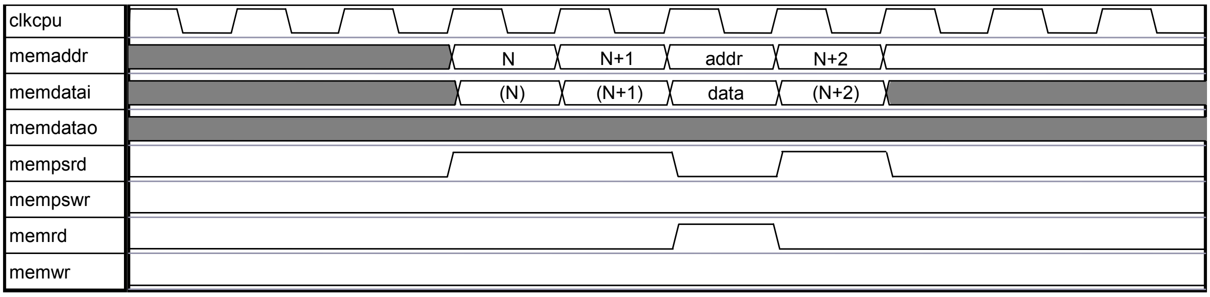


Figure 21. External Data Memory Read Cycle (Negedge SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of external data memory
	Data	- data read from address addr

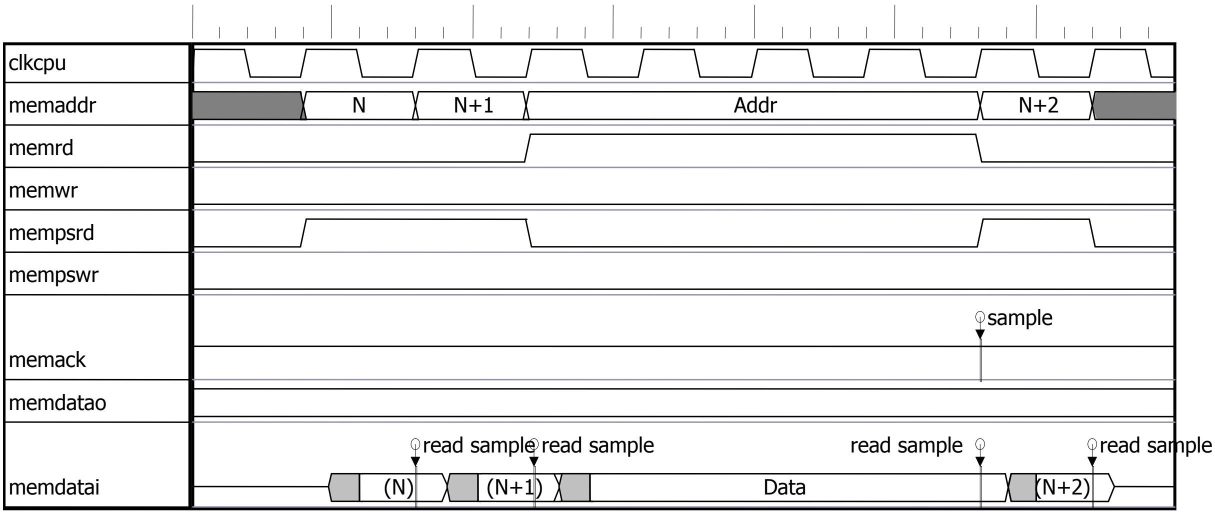


Figure 22. External Data Memory Read Cycle with 3 Wait States (Negedge SRAM)

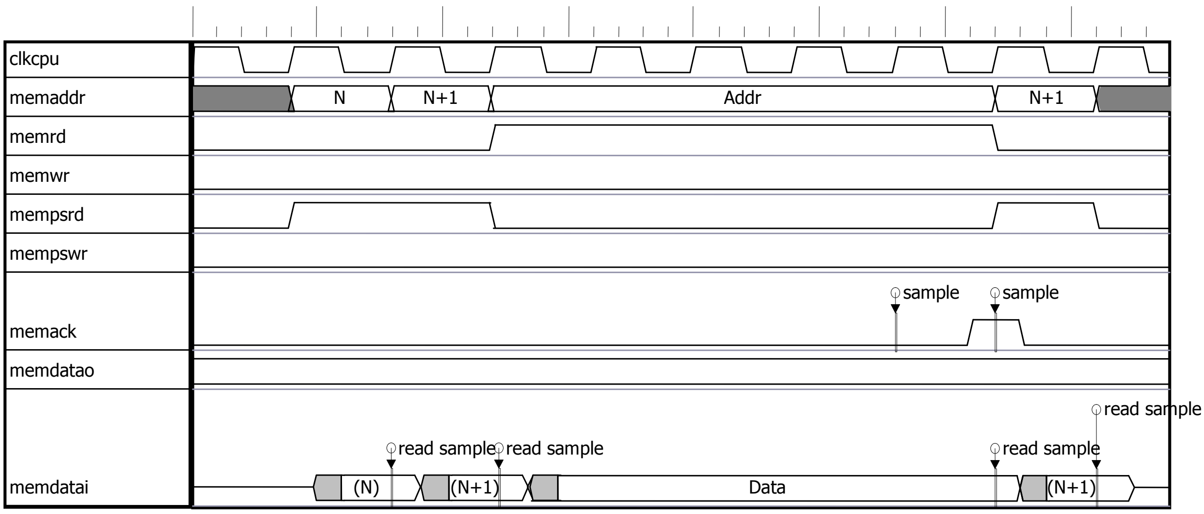


Figure 23. External Data Memory Read Cycle with 3 Wait States Delayed by Memack (Negedge SRAM)

2.5.7 External Data Memory Read Cycle (Posedge SRAM)

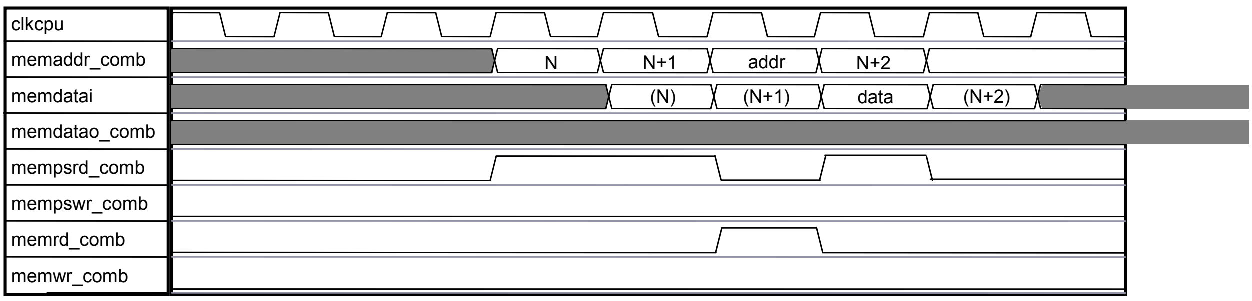


Figure 24. External Data Memory Read Cycle (Posedge SRAM)

Note: N - address of instruction performing a read

(N)	- instruction fetched from address N
N+1	- address of next instruction
Addr	- address of external data memory
Data	- data read from address addr

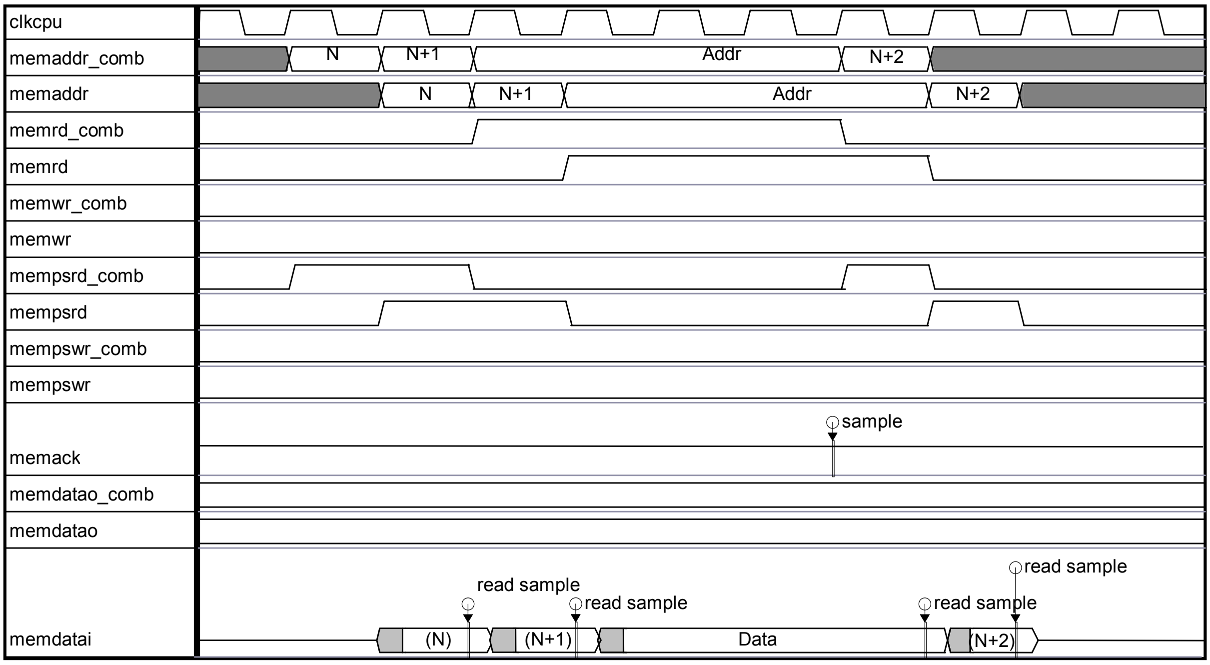


Figure 25. External Data Memory Read Cycle with 3 Wait States (Posedge memory)

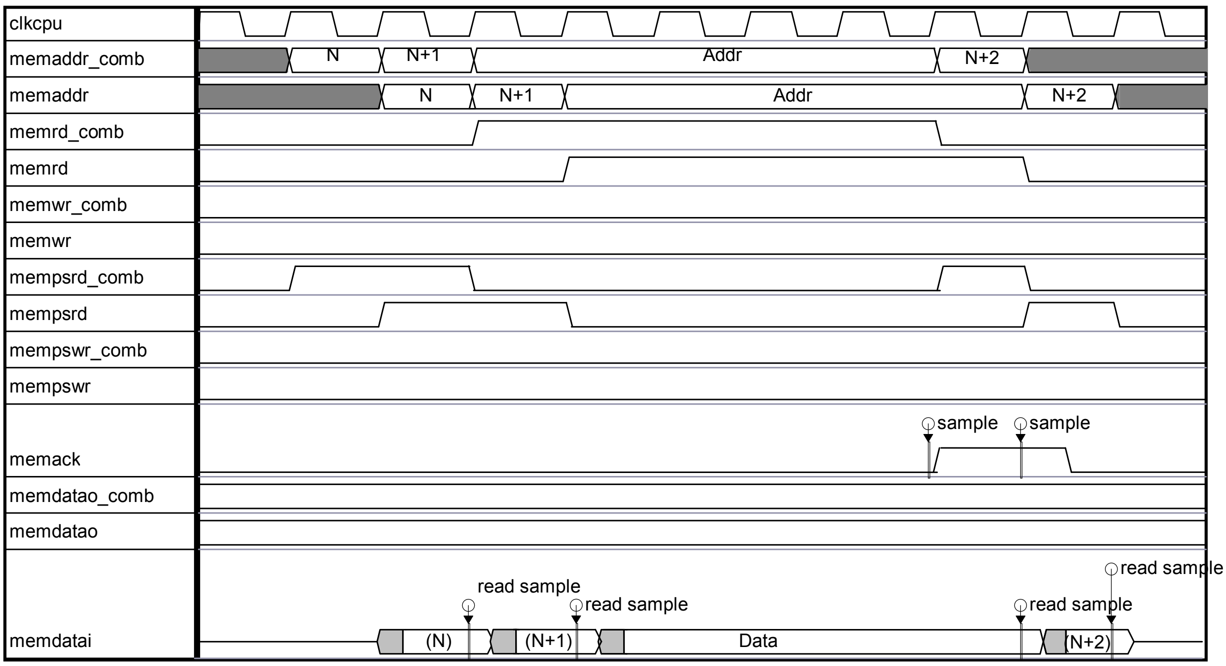


Figure 26. External Data Memory Read Cycle with 3 Wait States Delayed by Memack (Posedge memory)

2.5.8 External Data Memory Write Cycle (Negedge SRAM)

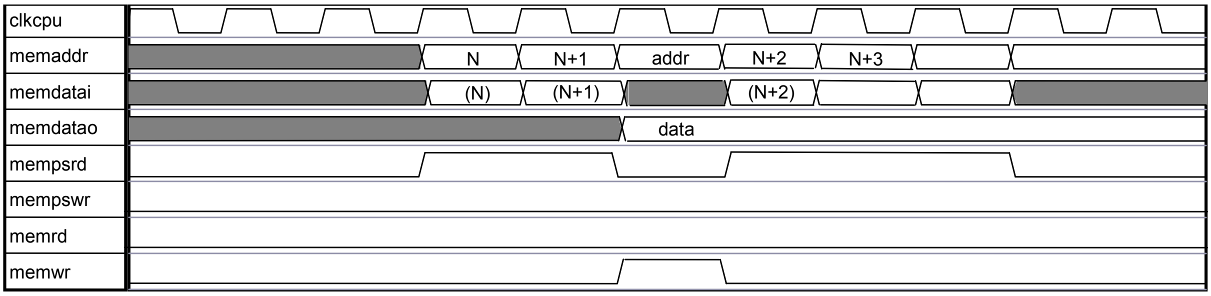


Figure 27. External Data Memory Write Cycle (Negedge SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of data memory being written
	Data	- data written at addr

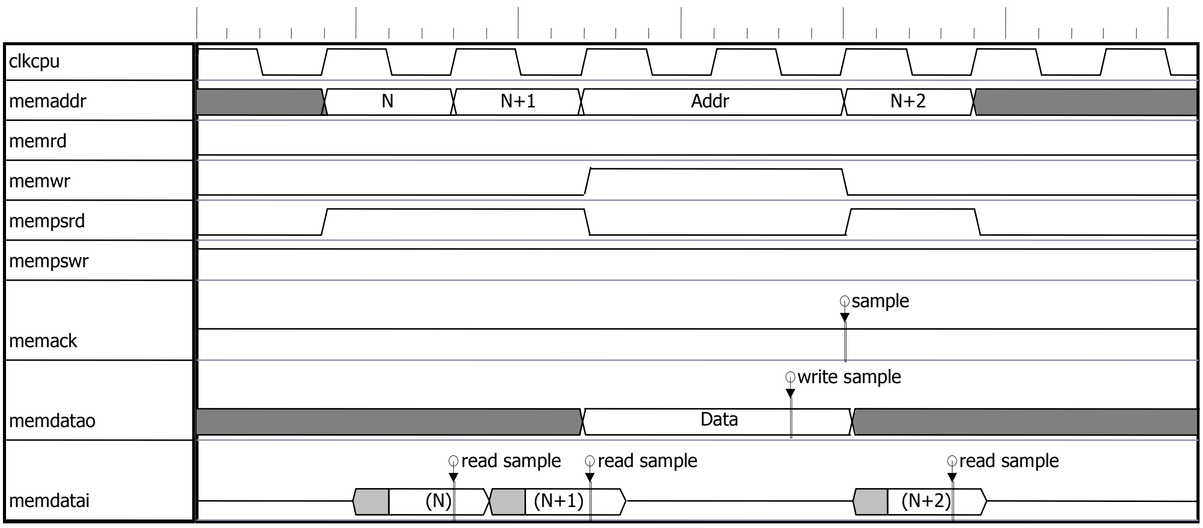


Figure 28. External Data Memory Write Cycle with 1 Wait State (Negedge SRAM)

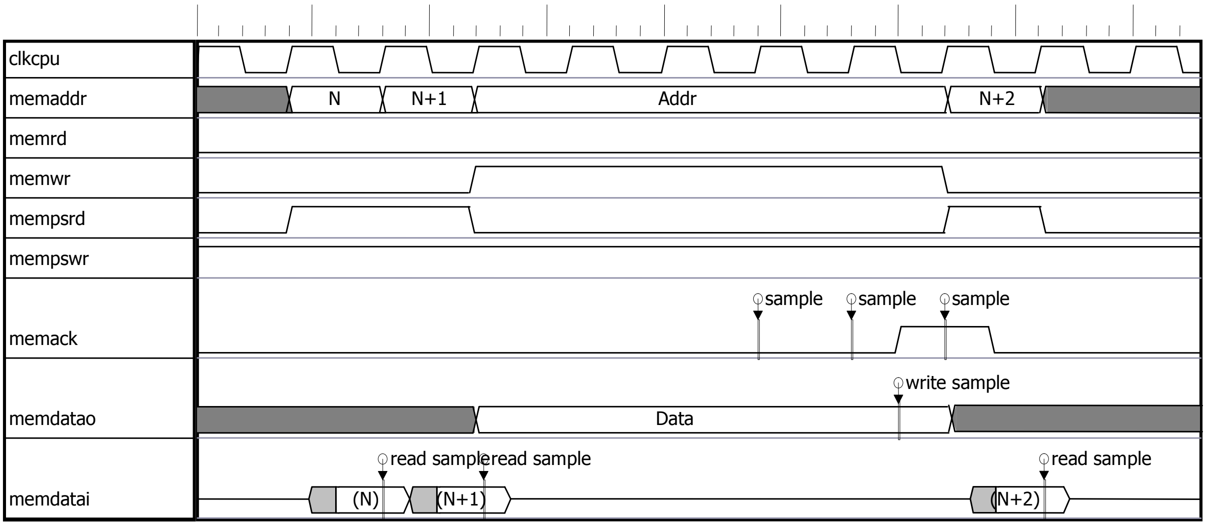


Figure 29. External Data Memory Write Cycle with 2 Wait States Delayed by Memack (Negedge SRAM)

2.5.9 External Data Memory Write Cycle (Posedge SRAM)

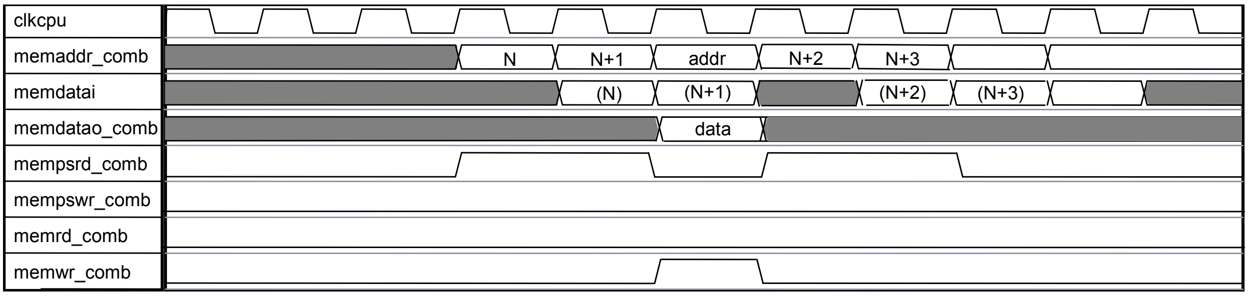


Figure 30. External Data Memory Write Cycle (Posedge SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of data memory being written
	Data	- data written at addr

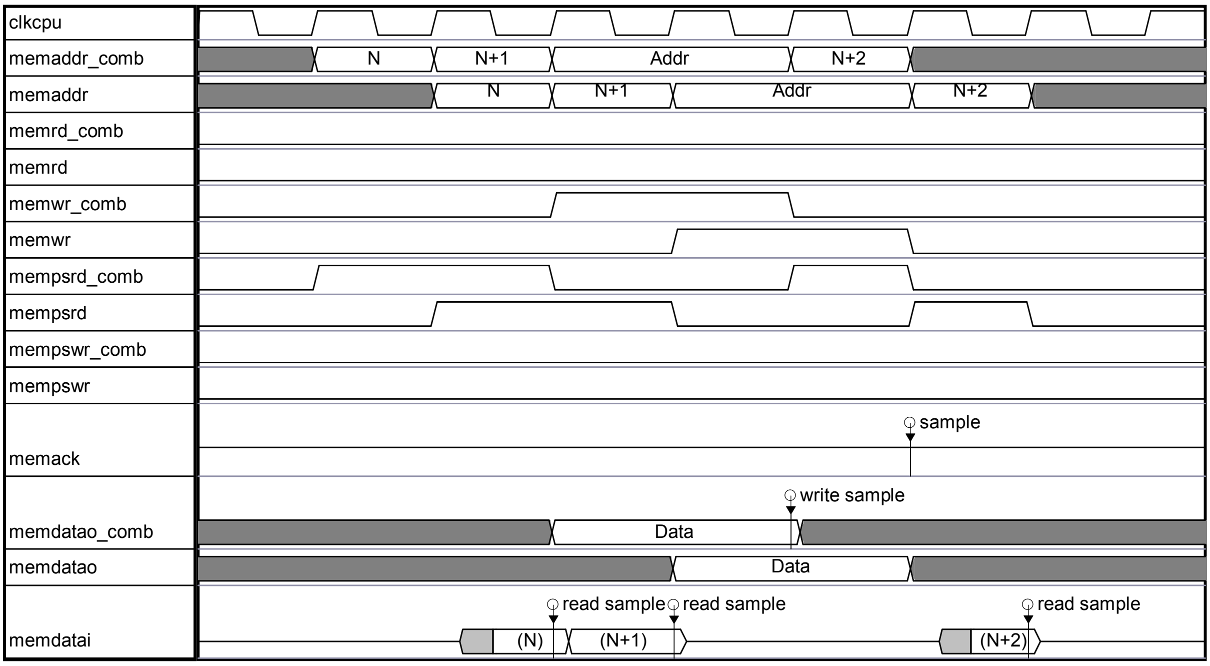


Figure 31. External Data Memory Write Cycle with 1 Wait State (Posedge SRAM)

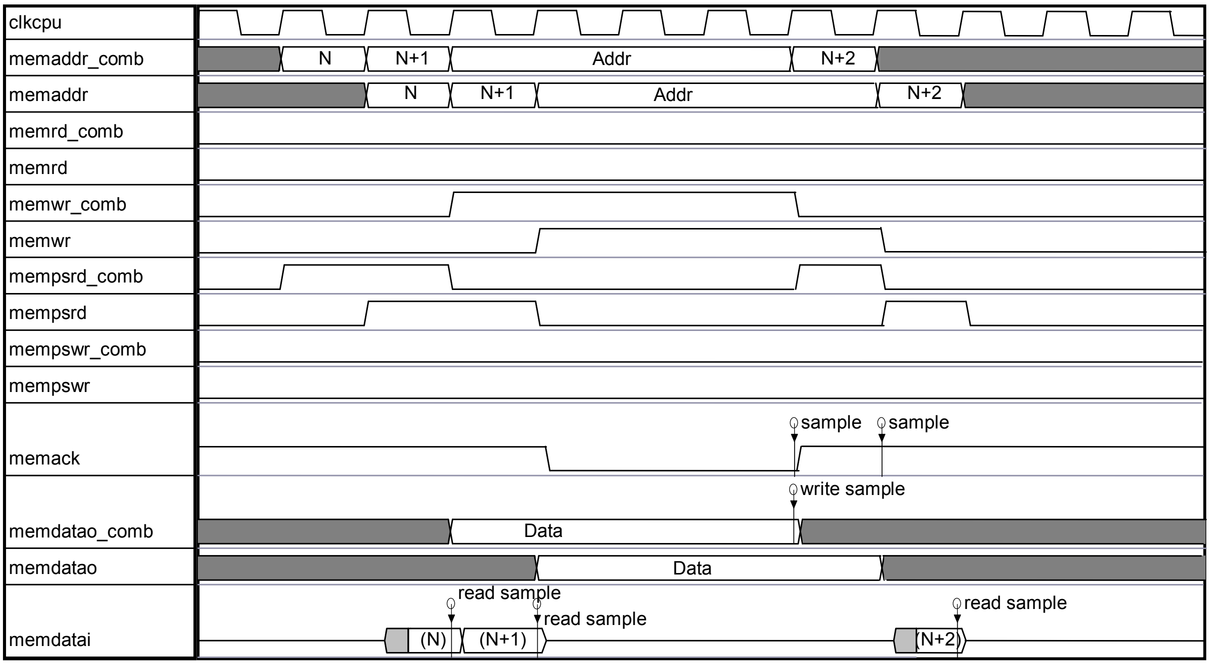


Figure 32. External Data Memory Write Cycle with 2 Wait States Delayed by Memack (Posedge SRAM)

2.5.10 External Data Memory Dword Read Cycle (Negedge SRAM)

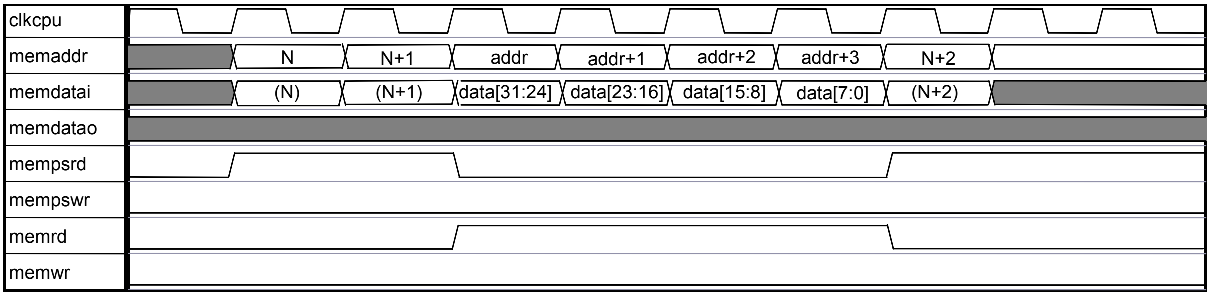


Figure 33. External Data Memory Dword Read Cycle (Negedge SRAM)

Note: N - address of instruction
(N) - instruction fetched from address N
N+1 - address of next instruction
Addr - address of external data memory
Data - data read from address addr

2.5.11 External Data Memory Dword Read Cycle (Posedge SRAM)

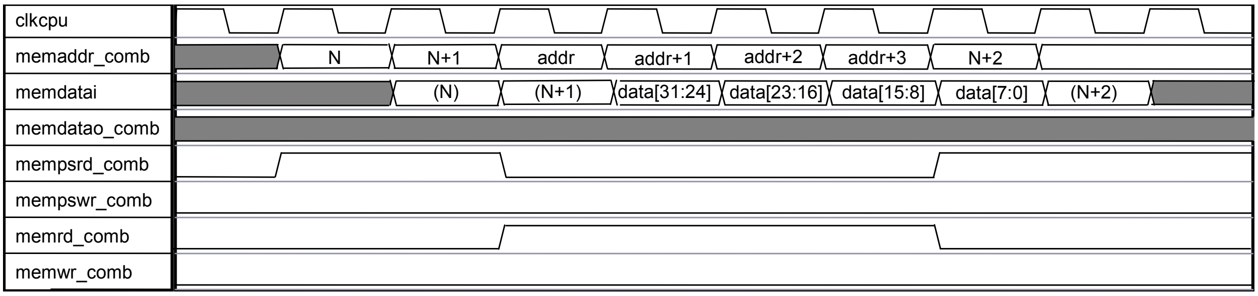


Figure 34. External Data Memory Dword Read Cycle (Posedge SRAM)

Note: N - address of instruction
(N) - instruction fetched from address N
N+1 - address of next instruction
Addr - address of external data memory
Data - data read from address addr

2.5.12 External Data Memory Dword Write Cycle (Negedge SRAM)

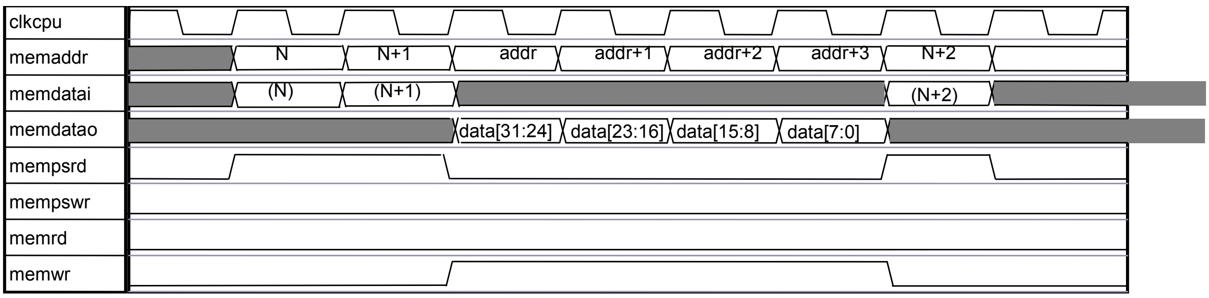


Figure 35. External Data Memory Dword Write Cycle (Negedge SRAM)

Note: N - address of instruction
(N) - instruction fetched from address N

N+1 - address of next instruction
Addr - address of external data memory
Data - data write to address addr

2.5.13 External Data Memory Dword Write Cycle (Posedge SRAM)

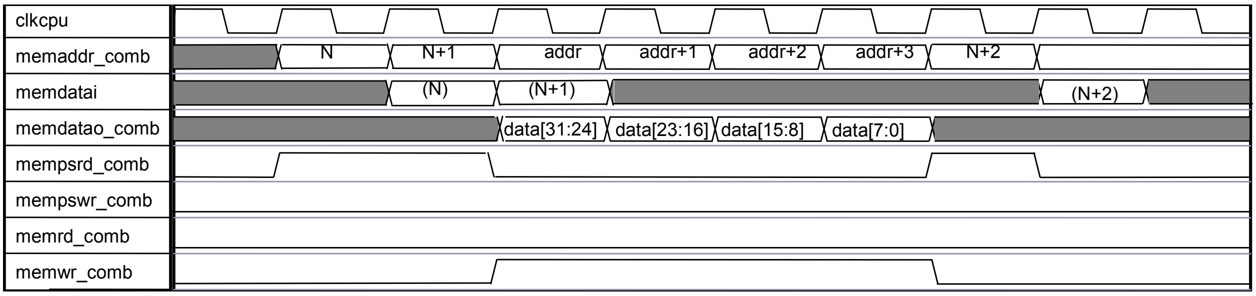


Figure 36. External Data Memory Dword Write Cycle (Posedge SRAM)

Note: N - address of instruction
(N) - instruction fetched from address N
N+1 - address of next instruction
Addr - address of external data memory
Data - data write to address addr

2.5.14 External Data Memory Word Read Cycle (Negedge SRAM)

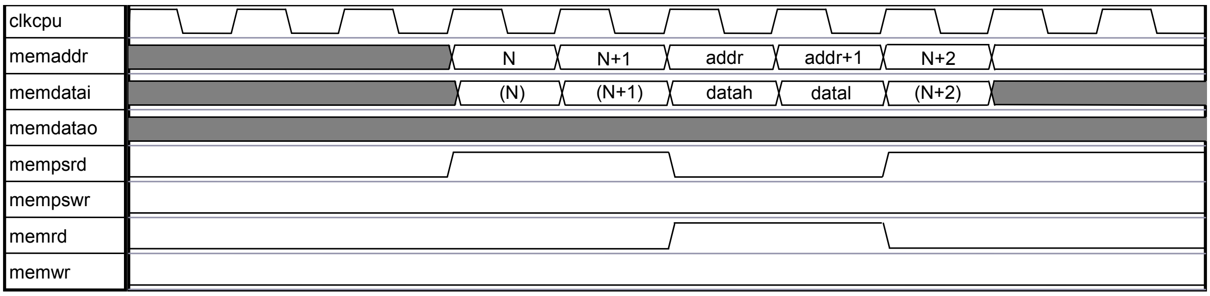


Figure 37. External Data Memory Word Read Cycle (Negedge SRAM)

Note: N - address of instruction
(N) - instruction fetched from address N
N+1 - address of next instruction
Addr - address of external data memory
Data - data read from address addr

2.5.15 External Data Memory Word Read Cycle (Posedge SRAM)

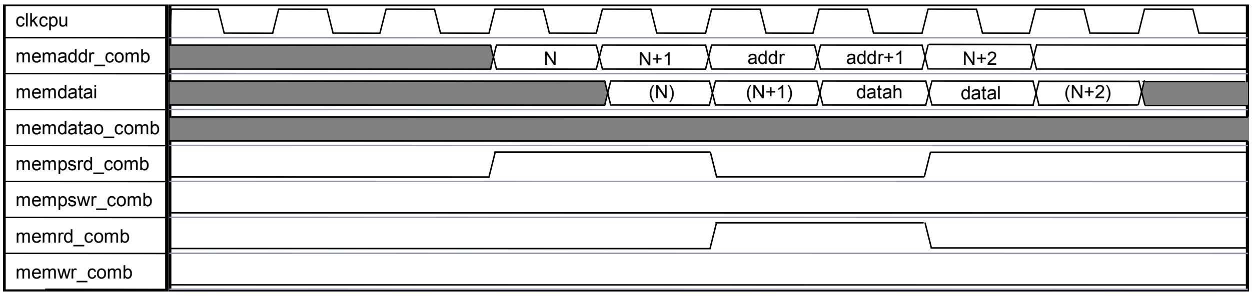


Figure 38. External Data Memory Word Read Cycle (Posedge SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of external data memory
	Data	- data read from address addr

2.5.16 External Data Memory Word Write Cycle (Negedge SRAM)

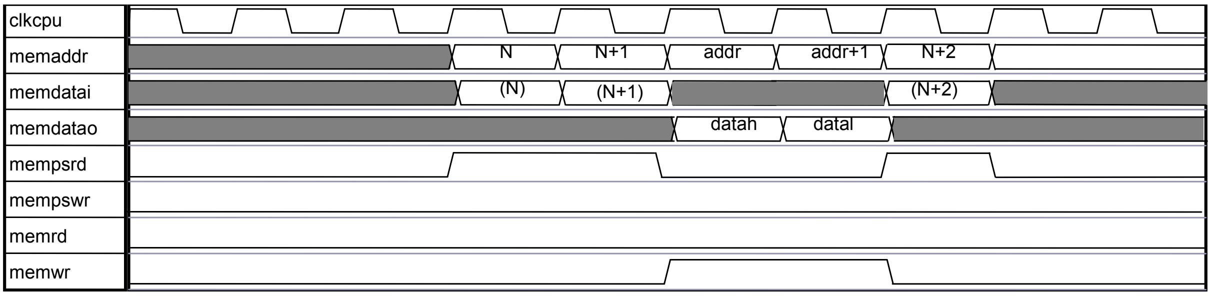


Figure 39. External Data Memory Word Write Cycle (Negedge SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of external data memory
	Data	- data write to address addr

2.5.17 External Data Memory Word Write Cycle (Posedge SRAM)

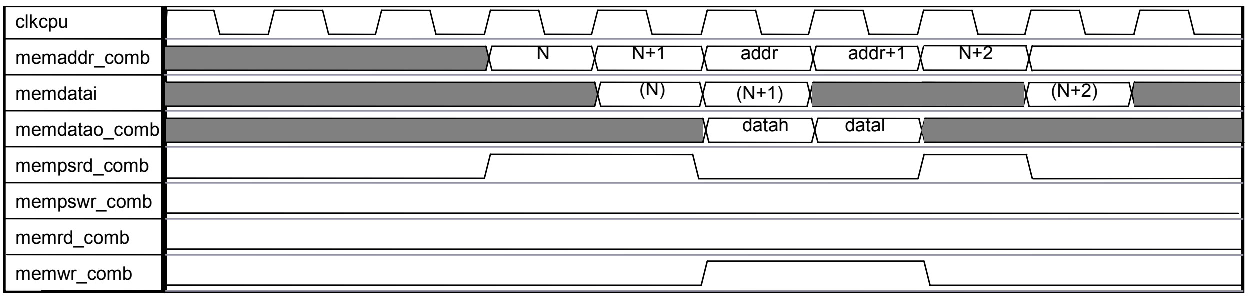


Figure 40. External Data Memory Word Write Cycle (Posedge SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N

N+1 - address of next instruction
Addr - address of external data memory
Data - data write to address addr

2.6. Internal Program Memory Interface (R80251XC-I(F) only)

2.6.1 Interface Description

The R80251XC-I(F) contains interface to Internal Program Memory.

The interface consists of the configurable width (13-bit by default) address bus "romaddr", the 8-bit input data bus "romdatai", the 8-bit output data bus "romdatao", and control signals "romrd", "romwr". This interface should be used to access both rising- or falling-edge triggered synchronous RAM/ROM, or asynchronous RAM if the write strobe is gated by the 'clkcpu' to ensure proper timing, or synchronous ROM. As each access takes at least two clock cycles, both rising or falling edge triggered memories can be used.

The Internal Program Memory Interface is mapped in the :FF page at address 0xFF:0000 when the 'ea' input is driven high. If the 'ea' is low, the Internal Program Memory is inaccessible.

The Internal Program Memory Interface provides software controlled wait states. The number of software controlled waitstates is defined by the UCONFIG0 and UCONFIG1 registers. An access can only be finished when the internal (software-controlled) wait state has finished.

2.6.2 Internal Program Memory Read Cycle (SRAM/ROM)

The figure below shows an example read of the instruction at address N.

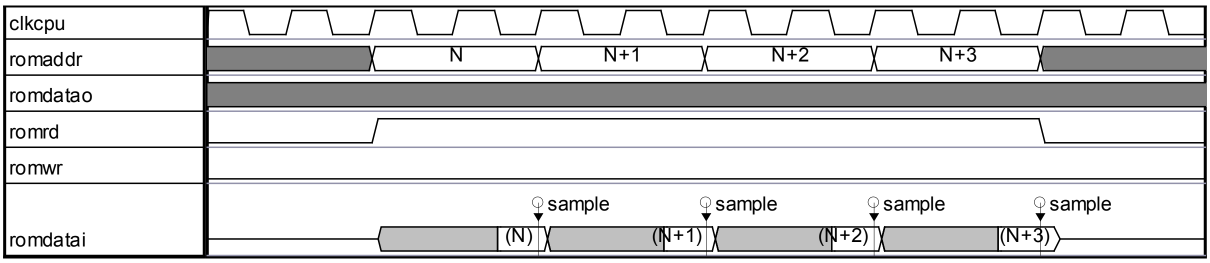


Figure 41. Internal Program Memory Read Cycle (SRAM/ROM)

Note: N - address of an instruction
(N) - instruction fetched from address N

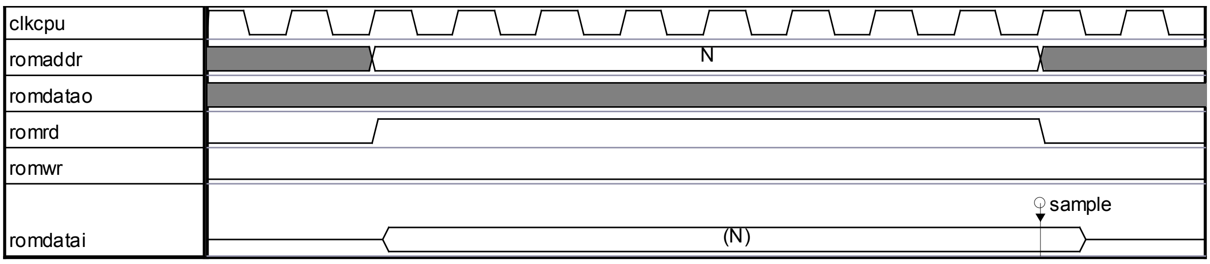


Figure 42. Program Memory Read Cycle with 3 Wait States

2.6.3 Program Memory Write Cycle (SRAM)

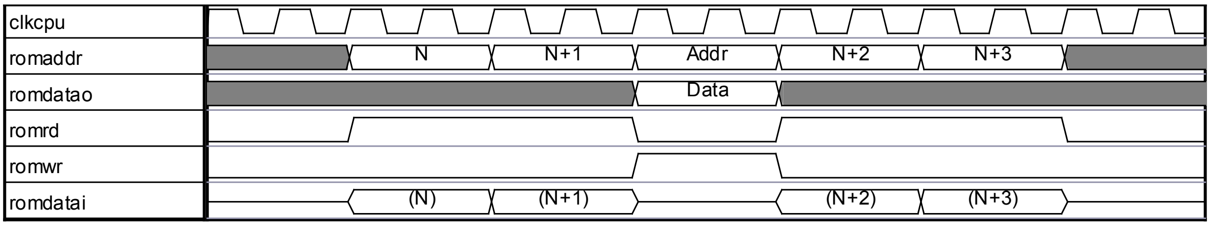


Figure 43. Internal Program Memory Write Cycle (SRAM)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of program memory being written
	Data	- data written at 'Addr'

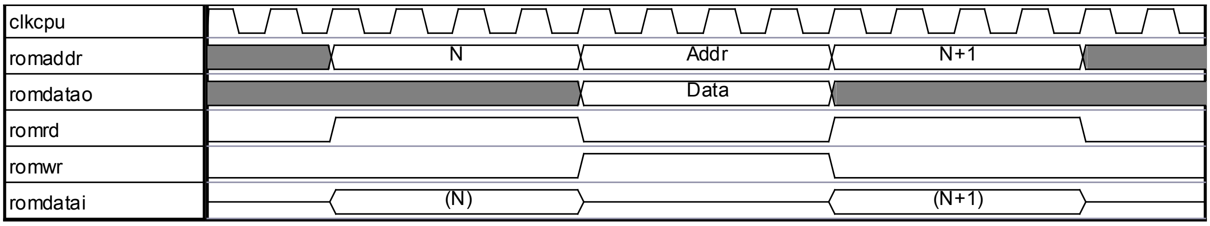


Figure 44. Program Memory Write Cycle with 1 Wait State (SRAM)

2.7. External Memory Interface (R80251XC-I(F) only)

2.7.1 Interface Description

The R80251XC contains the multiplexed interface to External Memory which is used to access external data memory and program memory.

The interface consists of the Port0, Port2 and Port3, plus 'ale' and 'psen' signals. The R80251XC-I(F) version is limited to 18 bits of address bus available to the outside, as in the original Intel devices.

The External Memory Interface provides software controlled wait states. The number of software controlled waitstates is defined by the UCONFIG0 and UCONFIG1 registers. Note that one wait state always takes two clock cycles.

The functionality of the External Memory Interface depends on the UCONFIG0 and UCONFIG1 registers. For details, see section 2.4.54 .

The Code is always fetched with the 'psen' signal (low active). The memory is always written with the 'port3o[6]' signal (wr – write enable output). The Data Memory is read with the 'port3o[7]' signal (rd – read enable output).

2.7.2 Program Memory Read Cycle (R80251XC-I(F) only)

The figure below shows an example read of the instruction at address N in non-page mode (UCONFIG0.1 = 1), standard ALE (UCONFIG0.4 = 1) and no wait states.

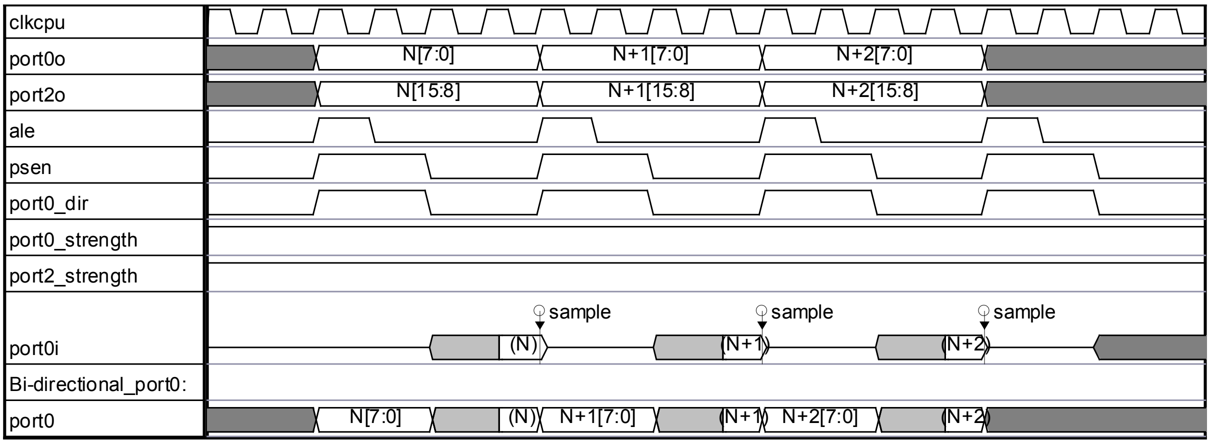


Figure 45. Program Memory Read Cycle (R80251XC-I(F) version)

Note:	N	- address of an instruction
	(N)	- instruction fetched from address N

2.7.3 External Data Memory Read Cycle (R80251XC-I(F) only)

The figure below shows an example one byte data read in non-page mode (UCONFIG0.1 = 1), standard ALE (UCONFIG0.4 = 1) and no wait states.

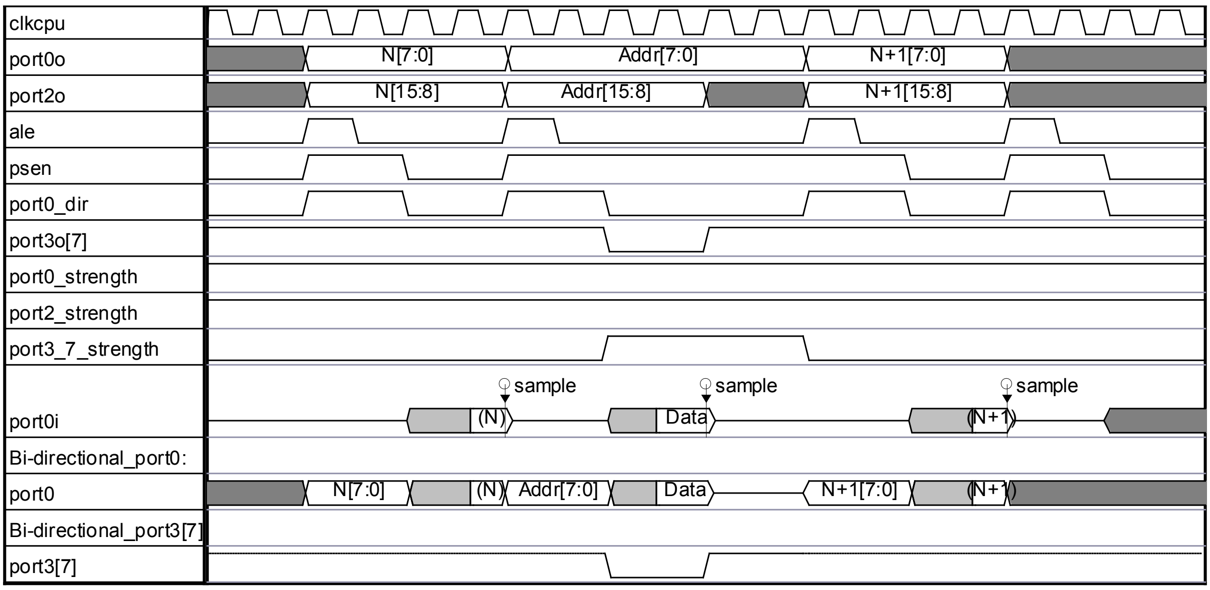


Figure 46. External Data Memory Read Cycle (R80251XC-I(F) version)

Note:	N	- address of an instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of memory being read
	Data	- data read from Addr

2.7.4 External Memory Write Cycle (R80251XC-I(F) only)

The figure below shows an example one byte write in non-page mode (UCONFIG0.1 = 1), standard ALE (UCONFIG0.4 = 1) and no wait states.

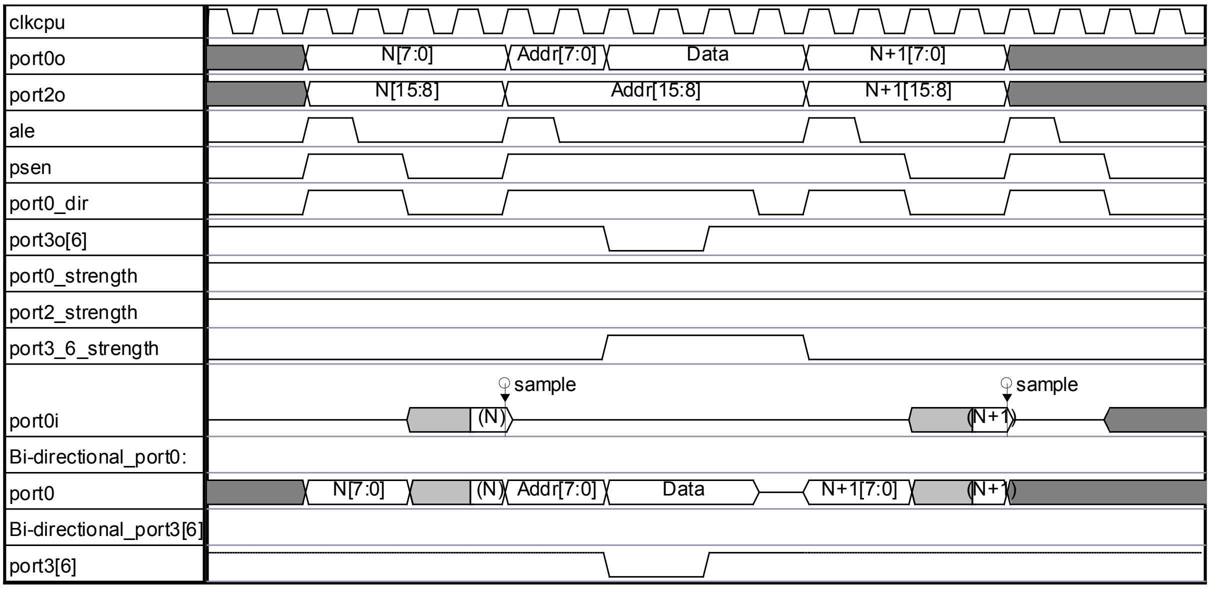


Figure 47. External Memory Write Cycle (R80251XC-I(F) version)

Note:	N	- address of an instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	Addr	- address of memory being written
	Data	- data written at Addr

2.8. Internal Data Memory Interface

2.8.1 Interface Description

The R80251XC provides interface to the microcontroller's Internal Data Memory (which can be implemented as off-core rising-edge triggered synchronous RAM or register file). Note that when the CPU encounters a wait state, or for the R80251XC-I(F) version, the single read/write cycle takes multiple cycles, depending on the number of wait states, or two for the R80251XC-I(F).

2.8.2 Internal Data Memory Read Cycle

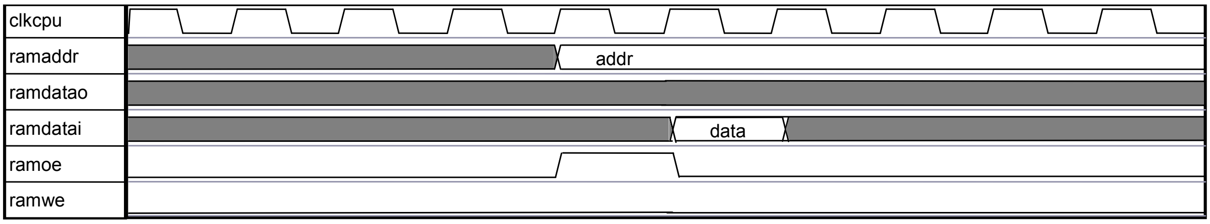


Figure 48. Internal Data Memory Read Cycle

Note: Addr - address of internal data memory
Data - data read from address addr

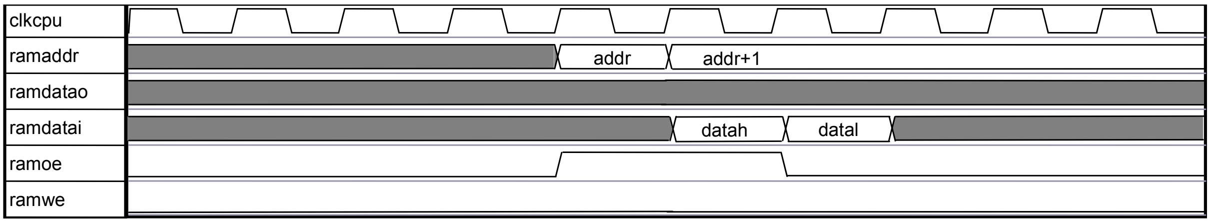


Figure 49. Internal Data Memory Read Cycle (Word Read)

Note: Addr - address of internal data memory
Data - data read from address addr

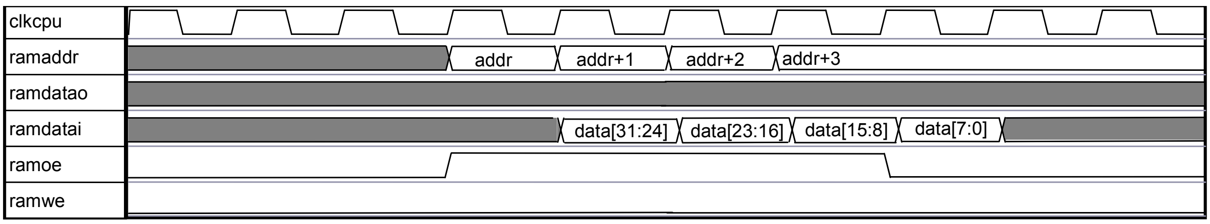


Figure 50. Internal Data Memory Read Cycle (Dword Read)

Note: Addr - address of internal data memory
Data - data read from address addr

2.8.3 Internal Data Memory Write Cycle

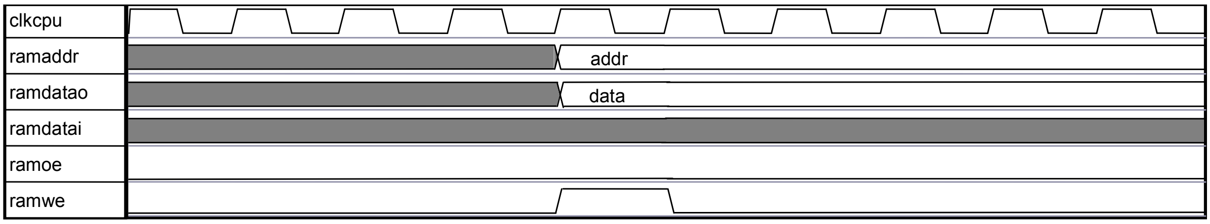


Figure 51. Internal Data Memory Write Cycle

Note: Addr - address of internal data memory
Data - data written into address addr

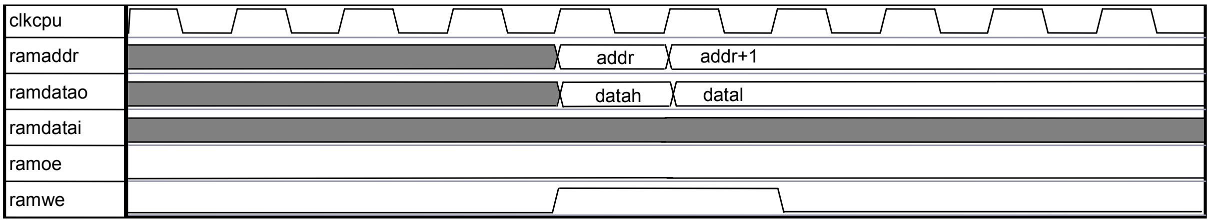


Figure 52. Internal Data Memory Write Cycle (Word Write)

Note: Addr - address of internal data memory
Data - data written into address addr

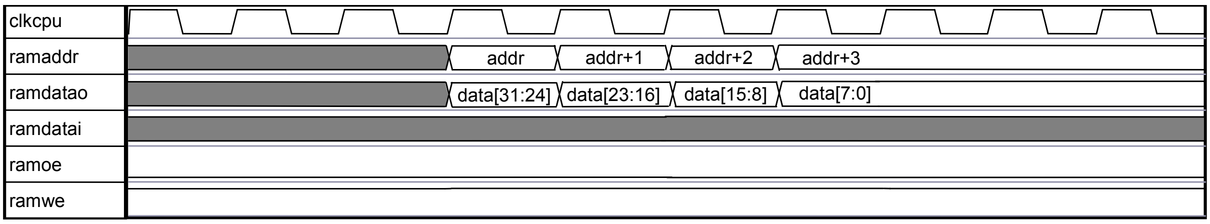


Figure 53. Internal Data Memory Write Cycle (Dword Write)

Note: Addr - address of internal data memory
Data - data written into address addr

It is possible that two consecutive write operations are performed to different addresses. It happens when a subroutine or interrupt call is executed and the Program Counter is pushed onto stack.

2.9. External Special Function Registers Interface

2.9.1 Interface Description

The R80251XC provides interface to External Special Function Registers (external to the microcontroller core).

Note that when the CPU encounters a wait state, or for the R80251XC-I(F) version, the single read/write cycle takes multiple cycles, depending on the number of wait states, or two for the R80251XC-I(F).

This interface can be easily adapted via an external wrapper to the AMBA™, VCI™ or OCP™ interfaces.

2.9.2 External SFR Read Cycle

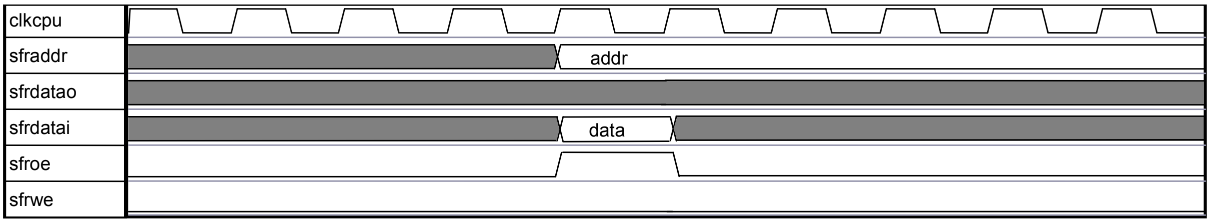


Figure 54. External Special Function Register Read Cycle

Note: Addr - address of special function register
Data - data read from address addr

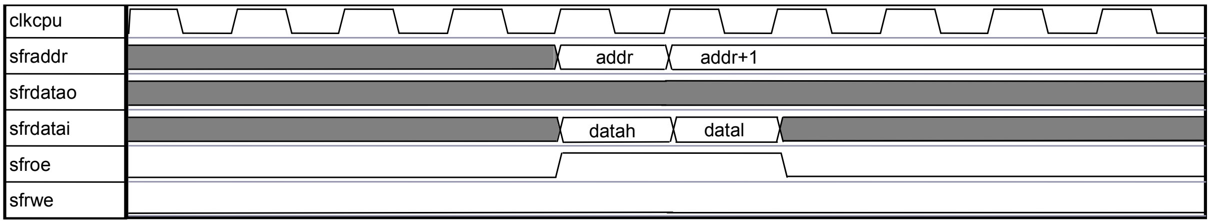


Figure 55. External Special Function Register Read Cycle (Word Read)

Note: Addr - address of special function register
Data - data read from address addr

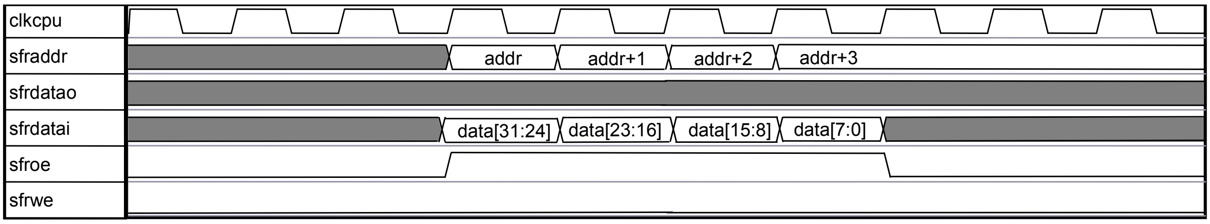


Figure 56. External Special Function Register Read Cycle (Dword Read)

Note: Addr - address of special function register
Data - data read from address addr

2.9.3 External SFR Write Cycle

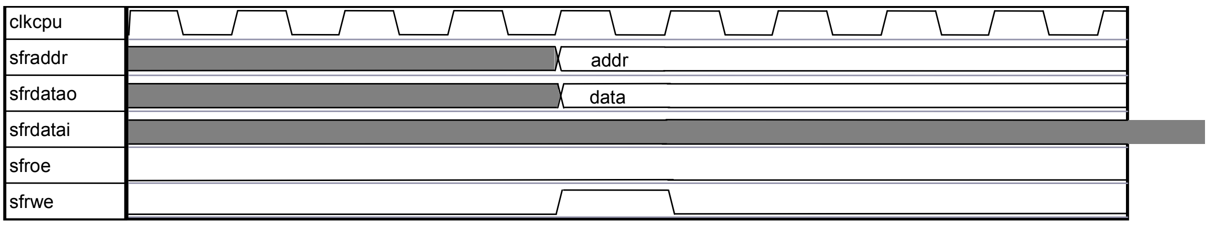


Figure 57. External Special Function Registers Write Cycle

Note: Addr - address of special function register
Data - data written into address Addr

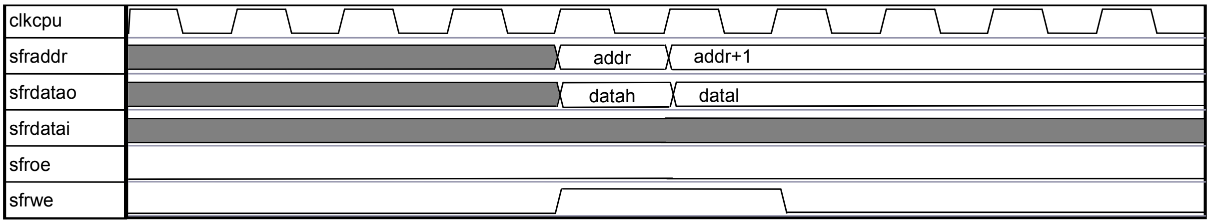


Figure 58. External Special Function Registers Write Cycle (Word)

Note: Addr - address of special function register
Data - data written into address Addr (Word)

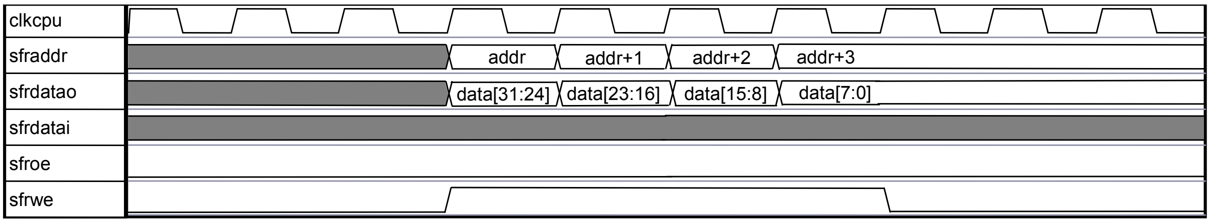


Figure 59. External Special Function Registers Write Cycle (Dword)

Note: Addr - address of special function register
Data - data written into address Addr (Dword)

2.10. Special Case

Read and write are similar, so this section contains only examples of read cycle.

2.10.1 Access to Sfrs (3 Bytes) and Internal RAM (1 Byte) - Dword

Example for this case is MOV Drk,dir8 where dir8 = 7DH

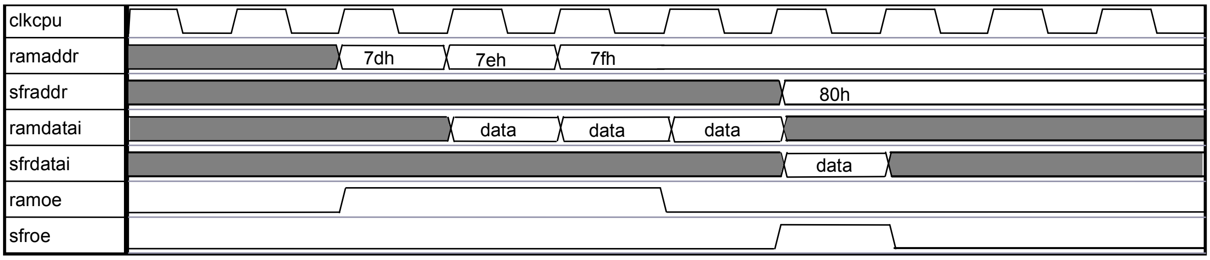


Figure 60. Access With Dir8 Addressing to SFR and IRAM

2.10.2 Access to Sfrs (2 Bytes) and Internal RAM (2 Byte) - Dword

Example for this case is MOV Drk,dir8 where dir8 = 7EH

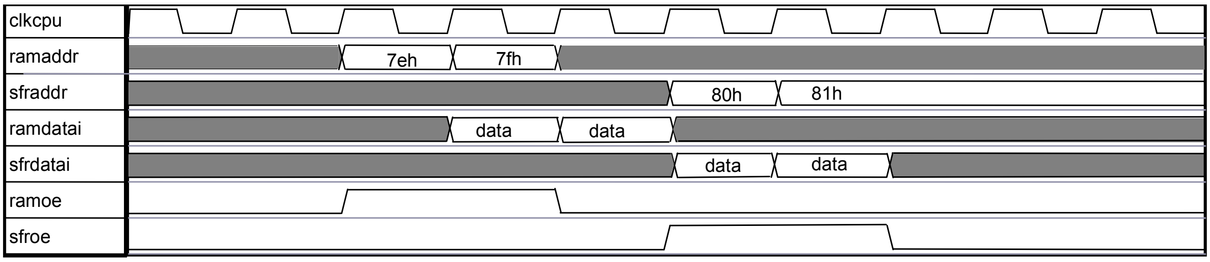


Figure 61. Access With Dir8 Addressing to SFR and IRAM

2.10.3 Access to Sfrs (1 Bytes) and Internal RAM (3 Byte) – Dword

Example for this case is MOV Drk,dir8 where dir8 = 7FH

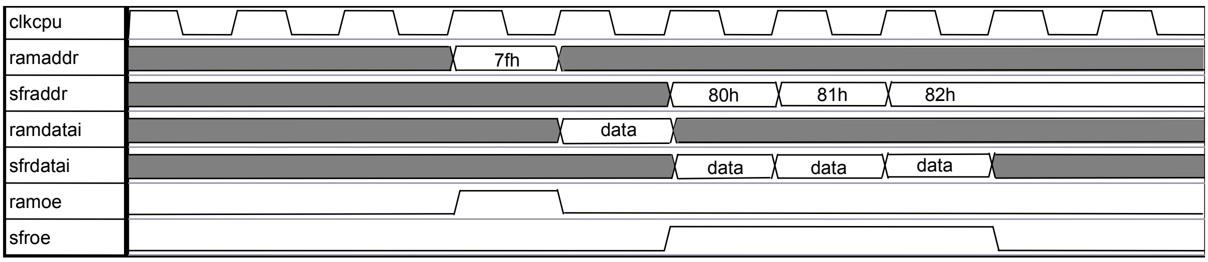


Figure 62. Access With Dir8 Addressing to SFR and IRAM

2.10.4 Access to Sfrs (1 Bytes) and Internal RAM (1 Byte) – Word

Example for this case is MOV Wrj,dir8 where dir8 = 7FH

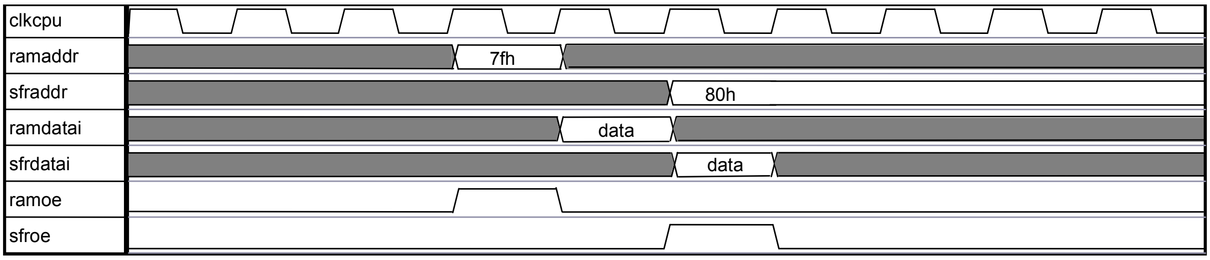


Figure 63. Access With Dir8 Addressing to SFR and IRAM

2.10.5 Access to Internal RAM (1 Byte) and External RAM (1 Byte) – Word

Example for this case is MOV Wrj,dir16 where dir16 = 41FH

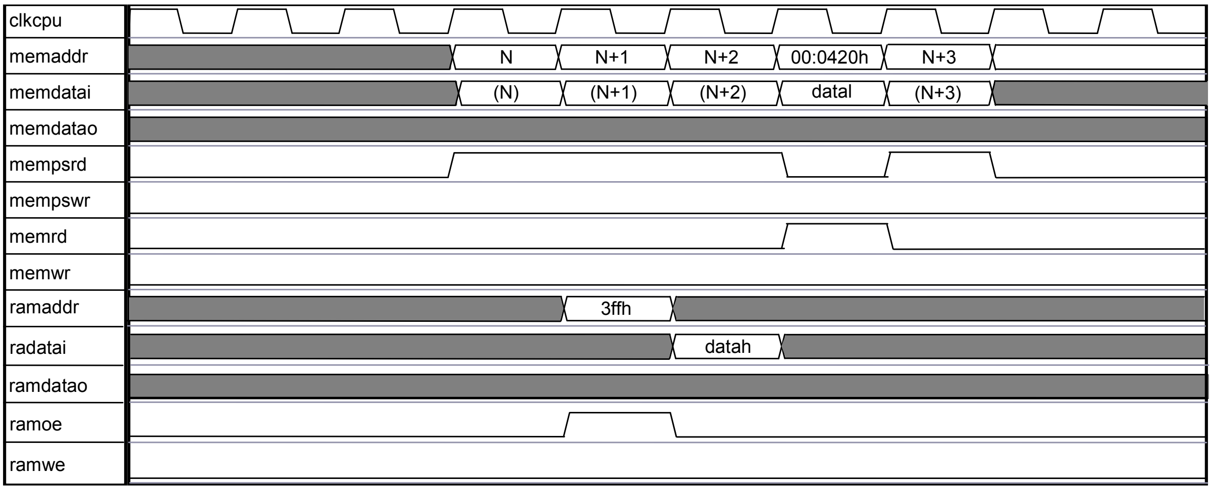


Figure 64. Access to IRAM and External RAM (Word)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	3FFH	- address of internal data memory (logical address is 41FH)
	datah	- data read from address 3FFH from IRAM
	datal	- data read from address 00:0420H from external memory

2.10.6 Access to IRAM (2 Bytes) and External RAM (2 Bytes) – Dword

Example for this case is MOV Drk,dir16 where dir16 = 41EH

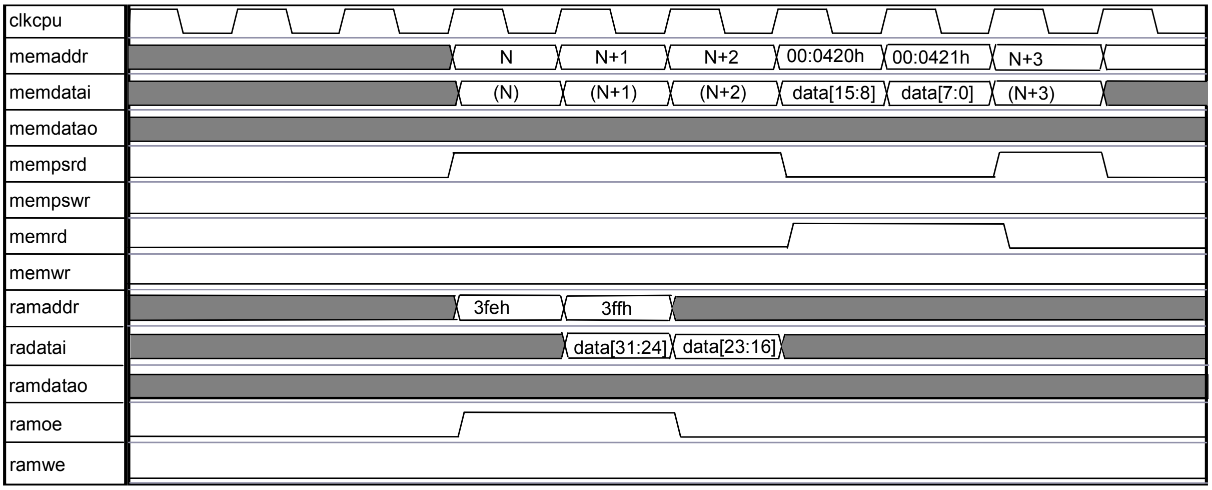


Figure 65. Access to IRAM and External RAM (Dword)

Note:	N	- address of instruction
	(N)	- instruction fetched from address N
	N+1	- address of next instruction
	data	- data read from IRAM (3FEH, 3FFH) or external memory (00:0420H and 00:0421H)

* Example: ramaddr = 3FEH when logical address is 41EH

2.11. On-Chip Debug Support Interface

2.11.1 Interface Description

The R80251XC contains the interface to the On-Chip Debug Support compliant with IEEE1149.1 standard. Detailed description can be found in section OCDS 5.1.3 e).

2.12. Serial Interfaces

2.12.1 Serial 0 and Serial 1

The R80251XC contains up to two Serial Port Interfaces (UARTs). For details refer to the description of Serial Port 0 (5.8) and Serial Port 1 (5.9).

2.12.2 I2C Interface

The R80251XC contains an optional Inter-Integrated Circuit Bus (I2C™) Interface. For details refer to the full description of I2C component (5.17).

2.12.3 Secondary I2C Interface

The R80251XC contains an optional Secondary Inter-Integrated Circuit Bus (I2C™) Interface. It is identical (except for SFR locations & interrupt involved) to the primary I2C Interface, and can only be implemented when the primary interface is implemented too.

2.12.4 SPI Interface

The R80251XC contains an optional Perial Peripheral Interface (SPI). For details refer to the full description of SPI component (5.18).

2.13. Hold Interface

2.13.1 Interface Description

The R80251XC contains ythe HOLD interface. Detailed description can be found in section 5.1.3 e) (Hold Mode).

3. Programming Specification

3.1. Instruction Set

All R80251XC instructions are binary code compatible and perform the same functions as they do within the industry standard 8051. The following tables give a summary of instruction cycles of the R80251XC microcontroller core.

Table 55 and Table 56 contain notes on mnemonics used in Instruction Set tables.

Table 56 … Table 60 show instruction hexadecimal codes, numbers of bytes and machine cycles that each instruction takes to be executed. Note the number of cycles is given for no program memory wait states.

Table 55. Notation For Instruction Operands

Symbol	Description
Notation for Register Operands	
@Ri	A memory location (00H–FFH) addressed indirectly via byte register R0 or R1
Rn	Working register R0-R7
Rm	Byte register R0–R15 of the currently selected register file
Rmd	Destination register
Rms	Source register
Wrj	Word register WR0, WR2, ..., WR30 of the currently selected register file
WRjd	Destination register
WRjs	Source register
@Wrj	A memory location (00:0000H–00:FFFFH) addressed indirectly through word register WR0–WR30
@WRj+dis16	Data RAM location (00:0000H–00:FFFFH) addressed indirectly through a word register (WR0–WR30) + displacement value, where the displacement value is from 0 to 64 Kbytes.
DRk	Dword register DR0, DR4, ..., DR28, DR56, DR60 of the currently selected register file
DRkd	Destination register
DRks	Source register
@Drk	A memory location (00:0000H–FF:FFFFH) addressed Indirectly through dword register DR0–DR28, DR56, DR60
@DRk+dis24	Data RAM location (00:0000H–FF:FFFFH) addressed indirectly through a dword register (DR0–DR28, DR56, DR60) + displacement value, where the displacement value is from 0 to 64 Kbytes
Notation for Direct Addresses	
dir8	An 8-bit direct address. This can be a memory address (00:0000H–00:00FFH) or an SFR address (S:00H - S:FFH).

Symbol	Description
dir16	A 16-bit memory address (00:0000H–00:FFFFH) used in direct addressing.
Notation for Immediate Addressing	
#data	An 8-bit constant that is immediately addressed in an instruction.
#data16	A 16-bit constant that is immediately addressed in an instruction.
#0data16	A 32-bit constant that is immediately addressed in an instruction. Upper word is filled with zeros.
#1data16	A 32-bit constant that is immediately addressed in an instruction. Upper word is filled with ones.
#short	A constant, equal to 1, 2, or 4, that is immediately addressed in an instruction.
Notation for Bit Addressing	
bit51	A directly addressed bit (bit number = 00H–FFH) in memory or an SFR. Bits 00H–7FH are the 128 bits in byte locations 20H–2FH in the on-chip RAM. Bits 80H–FFH are the 128 bits in the 16 SFR's with addresses that end in 0H or 8H: S:80H, S:88H, S:90H, . . . , S:F0H, S:F8H.
bit	A directly addressed bit in memory locations 00:0000H–00:007FH or in any defined SFR. A binary representation of the bit number (0–7) within a byte.
Notation for Destinations in Control Instructions	
rel	SJMP and all conditional jumps include an 8-bit offset byte. Its range is +127/-128 bytes relative to the first byte of the following instruction
addr11	Destination address for ACALL or AJMP, within the same 2-Kbyte page of program memory as the first byte of the following instruction
addr16	Destination address for LCALL or LJMP, can be anywhere within the 64-Kbyte page of program memory address space
addr24	A 24-bit destination address. A destination can be anywhere within the 16-Mbyte address space.

3.1.1 Instructions in Functional Order

A shaded cell denotes an instruction in the MCS–251 architecture.

The number of instruction cycles is given as follows: A /B where.
A is the number of instruction cycles needed to execute instruction which is fetched from instruction queue register (number of cycles depends on dec_newinstr).
B is the number of instruction cycles needed to execute instruction which is fetched from External Memory (number of cycles depends on dec_newinstr and dec_phase_block).

The table below doesn't take the waitstates into consideration.

Table 56. Arithmetic Operations

Mnemonic	Description	Bytes Bin/Src	Cycles Bin/Src
ADD A,Rn	Add register to accumulator	1 / 2	1 / 2

Mnemonic	Description	Bytes Bin/Src	Cycles Bin/Src
ADD A,dir8	Add directly addressed data to accumulator	2 / 2	1 / 1
ADD A,@Ri	Add indirectly addressed data to accumulator	1 / 2	2 / 3
ADD A,#data	Add immediate data to accumulator	2 / 2	1 / 1
ADD Rmd,Rms	Add byte register to byte register	3 / 2	2 / 1
ADD WRjd,WRjs	Add word register to word register	3 / 2	3 / 2
ADD Drkd,DRks	Add dword register to dword register	3 / 2	5 / 4
ADD Rm,#data	Add immediate 8-bit data to byte register	4 / 3	3 / 2
ADD WRj,#data16	Add immediate 16-bit data to word register	5 / 4	4 / 3
ADD Drk,#0data16	Add 16-bit unsigned immediate data to dword register	5 / 4	6 / 5
ADD Rm,dir8	Add directly addressed data to byte register	4 / 3	3 / 2
ADD Wrj,dir8	Add directly addressed data to word register	4 / 3	4 / 3
ADD Rm,dir16	Add directly addressed data (64 K) to byte register	5 / 4	3 / 2
ADD Wrj,dir16	Add directly addressed data (64 K) to word register	5 / 4	4 / 3
ADD Rm,@WRj	Add indirectly addressed (64 K) data to byte register	4 / 3	3 / 2
ADD Rm,@DRk	Add indirectly addressed (16 M)data to byte register	4 / 3	4 / 3
ADDC A,Rn	Add register to accumulator with the carry flag	1 / 2	1 / 2
ADDC A,dir8	Add directly addressed data to accumulator with the carry flag	2 / 2	1 / 1
ADDC A,@Ri	Add indirectly addressed data to accumulator with the carry flag	1 / 2	2 / 3
ADDC A,#data	Add immediate data to accumulator with the carry flag	2 / 2	1 / 1
SUB Rmd,Rms	Subtract byte register from byte register	3 / 2	2 / 1
SUB WRjd,WRjs	Subtract word register from word register	3 / 2	3 / 2
SUB Drkd,DRks	Subtract dword register from dword register	3 / 2	5 / 4
SUB Rm,#data	Subtract immediate 8-bit data from byte register	4 / 3	3 / 2
SUB WRj,#data16	Subtract immediate 16-bit data from word register	5 / 4	4 / 3
SUB Drk,#0data16	Subtract 16-bit unsigned immediate data from dword register	5 / 4	6 / 5
SUB Rm,dir8	Subtract directly addressed data from byte register	4 / 3	3 / 2
SUB Wrj,dir8	Subtract directly addressed data from word register	4 / 3	4 / 3
SUB Rm,dir16	Subtract directly addressed data (64 K) from byte register	5 / 4	3 / 2
SUB Wrj,dir16	Subtract directly addressed data (64 K) from word register	5 / 4	4 / 3
SUB Rm,@WRj	Subtract indirectly addressed (64 K) data from byte register	4 / 3	3 / 2
SUB Rm,@DRk	Subtract indirectly addressed (16 M)data from byte register	4 / 3	4 / 3
SUBB A,Rn	Subtract register from accumulator with borrow	1 / 2	1 / 2
SUBB A,direct	Subtract directly addressed data from accumulator with borrow	2 / 2	1 / 1
SUBB A,@Ri	Subtract indirectly addressed data from accumulator with borrow	1 / 2	2 / 3
SUBB A,#data	Subtract immediate data from accumulator with borrow	2 / 2	1 / 1
INC A	Increment accumulator	1 / 1	1 / 1
INC Rn	Increment register	1 / 2	1 / 2
INC dir8	Increment directly addressed location	2 / 2	2 / 2
INC @Ri	Increment indirectly addressed location	1 / 2	3 / 4
INC DPTR	Increment data pointer	1 / 1	1 / 1

Mnemonic	Description	Bytes Bin/Src	Cycles Bin/Src
INC Rm,#short	Increment byte register by 1, 2, or 4	3 / 2	2 / 1
INC WRj,#short	Increment word register by 1, 2, or 4	3 / 2	2 / 1
INC DRk,#short	Increment dword register by 1, 2, or 4	3 / 2	4 / 3
DEC A	Decrement accumulator	1 / 1	1 / 1
DEC Rn	Decrement register	1 / 2	1 / 2
DEC direct	Decrement directly addressed location	2 / 2	2 / 2
DEC @Ri	Decrement indirectly addressed location	1 / 2	3 / 4
DEC Rm,#short	Decrement byte register by 1, 2, or 4	3 / 2	2 / 1
DEC WRj,#short	Decrement word register by 1, 2, or 4	3 / 2	2 / 1
DEC DRk,#short	Decrement dword register by 1, 2, or 4	3 / 2	4 / 3
MUL AB	Multiply A and B	1 / 1	5 / 5
MUL Rmd,Rms	Multiply byte register and byte register	3 / 2	6 / 5
MUL Wrjd, WRjs	Multiply word register and word register	3 / 2	12 / 11
DIV AB	Divide A by B	1 / 1	10 / 10
DIV Rmd,Rms	Divide byte register and byte register	3 / 2	11 / 10
DIV WRjd,WRjs	Divide word register and word register	3 / 2	21 / 20
DA A	Decimally adjust accumulator	1 / 1	1 / 1
CMP Rmd,Rms	Compare byte register with byte register	3 / 2	2 / 1
CMP WRjd,WRjs	Compare word register with word register	3 / 2	3 / 2
CMP DRkd,DRks	Compare dword register with dword register	3 / 2	5 / 4
CMP Rm,#data	Compare immediate 8-bit data with byte register	4 / 3	3 / 2
CMP WRj,#data16	Compare immediate 16-bit data with word register	5 / 4	4 / 3
CMP Drk,#0data16	Compare 16-bit zero-extended immediate data with dword register	5 / 4	6 / 5
CMP Drk,#1data16	Compare 16-bit one-extended immediate data with dword register	5 / 4	6 / 5
CMP Rm,dir8	Compare directly addressed data with byte register	4 / 3	3 / 2
CMP Wrj,dir8	Compare directly addressed data with word register	4 / 3	4 / 3
CMP Rm,dir16	Compare directly addressed data (64 K) with byte register	5 / 4	3 / 2
CMP Wrj,dir16	Compare directly addressed data (64 K) with word register	5 / 4	4 / 3
CMP Rm,@WRj	Compare indirectly addressed (64 K) data with byte register	4 / 3	3 / 2
CMP Rm,@DRk	Compare indirectly addressed (16 M)data with byte register	4 / 3	4 / 3

Table 57. Logic Operations

Mnemonic	Description	Bytes	Cycles
ANL A,Rn	AND (Logical AND) register to accumulator	1 / 2	1 / 2
ANL A,dir8	AND directly addressed data to accumulator	2 / 2	1 / 1
ANL A,@Ri	AND indirectly addressed data to accumulator	1 / 2	2 / 3
ANL A,#data	AND immediate data to accumulator	2 / 2	1 / 1
ANL dir8,A	AND accumulator to directly addressed location	2 / 2	2 / 2
ANL dir8,#data	AND immediate data to directly addressed location	3 / 3	3 / 3
ANL Rmd,Rms	AND byte register to byte register	3 / 2	2 / 1

Mnemonic	Description	Bytes	Cycles
ANL WRjd,WRjs	AND word register to word register	3 / 2	3 / 2
ANL Rm,#data	AND immediate 8-bit data to byte register	4 / 3	3 / 2
ANL WRj,#data16	AND immediate 16-bit data to word register	5 / 4	4 / 3
ANL Rm,dir8	AND directly addressed data to byte register	4 / 3	3 / 2
ANL Wrj,dir8	AND directly addressed data to word register	4 / 3	4 / 3
ANL Rm,dir16	AND directly addressed data (64 K) to byte register	5 / 4	3 / 2
ANL Wrj,dir16	AND directly addressed data (64 K) to word register	5 / 4	4 / 3
ANL Rm,@WRj	AND indirectly addressed (64 K) data to byte register	4 / 3	3 / 2
ANL Rm,@DRk	AND indirectly addressed (16 M) data to byte register	4 / 3	4 / 3
ORL A,Rn	OR (Logical OR) register to accumulator	1 / 2	1 / 2
ORL A,dir8	OR directly addressed data to accumulator	2 / 2	1 / 1
ORL A,@Ri	OR indirectly addressed data to accumulator	1 / 2	2 / 3
ORL A,#data	OR immediate data to accumulator	2 / 2	1 / 1
ORL dir8,A	OR accumulator to directly addressed location	2 / 2	2 / 2
ORL dir8,#data	OR immediate data to directly addressed location	3 / 3	3 / 3
ORL Rmd,Rms	OR byte register to byte register	3 / 2	2 / 1
ORL WRjd,WRjs	OR word register to word register	3 / 2	3 / 2
ORL Rm,#data	OR immediate 8-bit data to byte register	4 / 3	3 / 2
ORL WRj,#data16	OR immediate 16-bit data to word register	5 / 4	4 / 3
ORL Rm,dir8	OR directly addressed data to byte register	4 / 3	3 / 2
ORL Wrj,dir8	OR directly addressed data to word register	4 / 3	4 / 3
ORL Rm,dir16	OR directly addressed data (64 K) to byte register	5 / 4	3 / 2
ORL Wrj,dir16	OR directly addressed data (64 K) to word register	5 / 4	4 / 3
ORL Rm,@WRj	OR indirectly addressed (64 K) data to byte register	4 / 3	3 / 2
ORL Rm,@DRk	OR indirectly addressed (16 M) data to byte register	4 / 3	4 / 3
XRL A,Rn	Exclusive OR (Logical Exclusive OR) register to accumulator	1 / 2	1 / 2
XRL A,dir8	Exclusive OR directly addressed data to accumulator	2 / 2	1 / 1
XRL A,@Ri	Exclusive OR indirectly addressed data to accumulator	1 / 2	2 / 3
XRL A,#data	Exclusive OR immediate data to accumulator	2 / 2	1 / 1
XRL dir8,A	Exclusive OR accumulator to directly addressed location	2 / 2	2 / 2
XRL dir8,#data	Exclusive OR immediate data to directly addressed location	3 / 3	3 / 3
XRL Rmd,Rms	Exclusive OR byte register to byte register	3 / 2	2 / 1
XRL WRjd,WRjs	Exclusive OR word register to word register	3 / 2	3 / 2
XRL Rm,#data	Exclusive OR immediate 8-bit data to byte register	4 / 3	3 / 2
XRL WRj,#data16	Exclusive OR immediate 16-bit data to word register	5 / 4	4 / 3
XRL Rm,dir8	Exclusive OR directly addressed data to byte register	4 / 3	3 / 2
XRL Wrj,dir8	Exclusive OR directly addressed data to word register	4 / 3	4 / 3
XRL Rm,dir16	Exclusive OR directly addressed data (64 K) to byte register	5 / 4	3 / 2
XRL Wrj,dir16	Exclusive OR directly addressed data (64 K) to word register	5 / 4	4 / 3
XRL Rm,@WRj	Exclusive OR indirectly addressed (64 K) data to byte register	4 / 3	3 / 2
XRL Rm,@DRk	Exclusive OR indirectly addressed (16 M) data to byte register	4 / 3	4 / 3
CLR A	Clear accumulator	1 / 1	1 / 1
CPL A	Complement accumulator	1 / 1	1 / 1

Mnemonic	Description	Bytes	Cycles
RL A	Rotate accumulator left	1 / 1	1 / 1
RLC A	Rotate accumulator left through the carry flag	1 / 1	1 / 1
RR A	Rotate accumulator right	1 / 1	1 / 1
RRC A	Rotate accumulator right through the carry flag	1 / 1	1 / 1
SLL Rm	Shift byte register left	3 / 2	1 / 2
SLL WRj	Shift word register left	3 / 2	2 / 1
SRA Rm	Shift byte register right through the MSB	3 / 2	2 / 1
SRA WRj	Shift word register left through the MSB	3 / 2	2 / 1
SRL Rm	Shift byte register right	3 / 2	2 / 1
SRL WRj	Shift word register right	3 / 2	2 / 1
SWAP A	Swap nibbles within the accumulator	1 / 1	2 / 2

Table 58. Data Transfer Operations

Mnemonic	Description	Bytes	Cycles
MOV A,Rn	Move register to accumulator	1 / 2	1 / 2
MOV A,dir8	Move directly addressed data to accumulator	2 / 2	1 / 1
MOV A,@Ri	Move indirectly addressed data to accumulator	1 / 2	2 / 3
MOV A,#data	Move immediate data to accumulator	2 / 2	1 / 1
MOV Rn,A	Move accumulator to register	1 / 2	1 / 2
MOV Rn,dir8	Move directly addressed data to register	2 / 3	1 / 2
MOV Rn,#data	Move immediate data to register	2 / 3	1 / 2
MOV dir8,A	Move accumulator to direct	2 / 2	2 / 2
MOV dir8,Rn	Move register to direct	2 / 3	2 / 3
MOV dir8,dir8	Move directly addressed data to directly addressed location	3 / 3	3 / 3
MOV direct,@Ri	Move indirectly addressed data to directly addressed location	2 / 3	3 / 4
MOV direct,#data	Move immediate data to directly addressed location	3 / 3	3 / 3
MOV @Ri,A	Move accumulator to indirectly addressed location	1 / 2	3 / 4
MOV @Ri,dir8	Move directly addressed data to indirectly addressed location	2 / 3	3 / 4
MOV @Ri,#data	Move immediate data to indirectly addressed location	2 / 3	3 / 4
MOV DPTR,#data16	Load Data Pointer with a 16-bit immediate	3 / 3	2 / 2
MOV Rmd,Rms	Move byte register to byte register	3 / 2	2 / 1
MOV WRjd,WRjs	Move word register to word register	3 / 2	2 / 1
MOV Drkd,DRks	Move dword register to dword register	3 / 2	3 / 2
MOV Rm,#data	Move immediate 8-bit data to byte register	4 / 3	3 / 2
MOV WRj,#data16	Move immediate 16-bit data to word register	5 / 4	3 / 2
MOV Drk,#0data16	Move 16-bit zero-extended immediate data to dword register	5 / 4	5 / 4
MOV Drk,#1data16	Move 16-bit one-extended immediate data to dword register	5 / 4	5 / 4
MOV Drk,dir8	Move directly addressed data to dword register	4 / 3	6 / 5
MOV Drk,dir16	Move directly addressed (64 K) data to dword register	5 / 4	6 / 5

Mnemonic	Description	Bytes	Cycles
MOV Rm,dir8	Move directly addressed data to byte register	4 / 3	3 / 2
MOV Wrj,dir8	Move directly addressed data to word register	4 / 3	4 / 3
MOV Rm,dir16	Move directly addressed data (64 K) to byte register	5 / 4	3 / 2
MOV Wrj,dir16	Move directly addressed data (64 K) to word register	5 / 4	4 / 3
MOV Rm,@WRj	Move indirectly addressed (64 K) data to byte register	4 / 3	2 / 2
MOV Rm,@DRk	Move indirectly addressed (16 M)data to byte register	4 / 3	4 / 3
MOV WRjd,@WRjs	Move indirectly addressed (64 K) data to word register	4 / 3	4 / 3
MOV Wrj,@DRk	Move indirectly addressed (16 M) data to word register	4 / 3	5 / 4
MOV dir8,Rm	Move byte register to directly addressed data	4 / 3	4 / 3
MOV dir8,Wrj	Move word register to directly addressed data	4 / 3	5 / 4
MOV dir16,Rm	Move byte register to directly addressed data (64 K)	5 / 4	4 / 3
MOV dir16,Wrj	Move word register to directly addressed data (64 K)	5 / 4	5 / 4
MOV @Wrj,Rm	Move byte register to indirectly addressed (64 K) data	4 / 3	4 / 3
MOV @Drk,Rm	Move byte register to indirectly addressed (16 M) data	4 / 3	5 / 4
MOV @Wrjd,WRjs	Move word register to indirectly addressed (64 K) data	4 / 3	5 / 4
MOV @Drk,Wrj	Move word register to indirectly addressed (16 M) data	4 / 3	6 / 5
MOV dir8,DRk	Move dword register to directly addressed data	4 / 3	7 / 6
MOV dir16,DRk	Move dword register to directly addressed data (64 K)	5 / 4	7 / 6
MOV Rm,@WRj+dis16	Move indirectly addressed data with displacement value (64 K) to byte register	5 / 4	6 / 5
MOV WRj,@WRj+dis16	Move indirectly addressed data with displacement value (64 K) to word register	5 / 4	7 / 6
MOV Rm,@DRk+dis24	Move indirectly addressed data with displacement value (16 M) to byte register	5 / 4	7 / 6
MOV WRj,@DRk+dis24	Move indirectly addressed data with displacement value (16 M) to word register	5 / 4	8 / 7
MOV @Wrj+dis16,Rm	Move byte register to indirectly addressed data with displacement value (64 K)	5 / 4	6 / 5
MOV @Wrj+dis16,Wrj	Move word register to indirectly addressed data with displacement value (64 K)	5 / 4	7 / 6
MOV @DRk+dis24,Rm	Move byte register to indirectly addressed data with displacement value (16 M)	5 / 4	7 / 6
MOV @DRk+dis24,Wrj	Move word register to indirectly addressed data with displacement value (16 M)	5 / 4	8 / 7
MOVH Drk(hi),#data16	Move 16-bit immediate data into upper word of dword register	5 / 4	3 / 2
MOVS Wrj,Rm	Move byte register to word register with sign extension	3 / 2	2 / 1
MOVZ Wrj,Rm	Move byte register to word register with zero extension	3 / 2	2 / 1
MOVC A,@A+DPTR	Load accumulator with a code byte relative to DPTR	1 / 1	6 / 6
MOVC A,@A+PC	Load accumulator with a code byte relative to PC	1 / 1	6 / 6
MOVX A,@Ri	1) Move external RAM (8-bit addr.) to accumulator	1 / 2	4 / 5
MOVX A,@DPTR	1) Move external RAM (16-bit addr.) to accumulator	1 / 1	5 / 5
MOVX @Ri,A	1) Move accumulator to external RAM (8-bit addr.)	1 / 1	4 / 4
MOVX @DPTR,A	1) Move accumulator to external RAM (16-bit addr.)	1 / 1	5 / 5
PUSH dir8	Push directly addressed data onto stack	2 / 2	2 / 2

Mnemonic	Description	Bytes	Cycles
PUSH #data	Push immediate data onto stack	4 / 3	4 / 3
PUSH #data16	Push 16-bit data immediate onto stack	5 / 4	5 / 5
PUSH Rm	Push byte register onto stack	3 / 2	4 / 3
PUSH WRj	Push word register onto stack	3 / 2	6 / 5
PUSH DRk	Push dword register onto stack	3 / 2	10 / 9
POP dir8	Pop directly addressed location from stack	2 / 2	3 / 3
POP Rm	Pop byte register from stack	3 / 2	3 / 2
POP WRj	Pop word register from stack	3 / 2	5 / 4
POP DRk	Pop dword register from stack	3 / 2	9 / 8
XCH A,Rn	Exchange register with accumulator	1 / 2	3 / 4
XCH A,dir8	Exchange directly addressed location with accumulator	2 / 2	3 / 3
XCH A,@Ri	Exchange indirect RAM with accumulator	1 / 2	4 / 5
XCHD A,@Ri	Exchange low-order nibbles of indirect and accumulator	1 / 2	4 / 5

Note:

[1)] External memory addressed by instructions in the MCS 51 architecture is in the region specified by DPXL (reset value = 01H).

Table 59. Program Branches

Mnemonic	Description	Bytes	Cycles
ACALL addr11	Absolute subroutine call	2 / 2	9 / 9
ECALL @DRk	Extended subroutine call, indirect	3 / 2	12 / 11
ECALL adr24	Extended subroutine call	5 / 4	14 / 13
LCALL @WRj	Long subroutine call, indirect	3 / 2	9 / 8
LCALL addr16	Long subroutine call	3 / 3	9 / 9
ERET	Extended subroutine return	3 / 2	6 / 6
RET	Return from subroutine	1 / 1	10 / 9
RETI (1)	Return from interrupt	1 / 1	6 / 6
AJMP addr11	Absolute jump	2 / 2	3 / 3
EJMP addr24	Extended jump	5 / 4	6 / 5
EJMP @DRk	Extended jump, indirect	3 / 2	7 / 6
LJMP @WRj	Long jump, indirect	3 / 2	6 / 5
LJMP addr16	Long jump	3 / 3	4 / 4
SJMP rel	Short jump (relative address)	2 / 2	3 / 3
JMP @A+DPTR	Jump indirect relative to the DPTR	1 / 1	5 / 5
JC rel	Jump if the carry flag is set	2 / 2	4 / 4
JNC rel	Jump if the carry flag is not set	2 / 2	4 / 4
JB bit,rel	Jump if directly addressed bit is set	5 / 4	7 / 6
JB bit51,rel	Jump if directly bit of 8-bit addressed location is set	3 / 3	5 / 5
JNB bit,rel	Jump if directly addressed bit is not set	5 / 4	7 / 6
JNB bit51,rel	Jump if directly bit of 8-bit addressed location is not set	3 / 3	5 / 5
JBC bit,rel	Jump if directly addressed bit is set and clear bit	5 / 4	10 / 9
JBC bit51,rel	Jump if directly bit of 8-bit addressed location is set and clear bit	3 / 3	7 / 7

Mnemonic	Description	Bytes	Cycles
JZ rel	Jump if accumulator is zero	2 / 2	5 / 5
JNZ rel	Jump if accumulator is not zero	2 / 2	5 / 5
JE rel	Jump if equal	3 / 2	5 / 4
JNE rel	Jump if not equal	3 / 2	5 / 4
JG rel	Jump if greater than	3 / 2	5 / 4
JLE rel	Jump if less than or equal	3 / 2	5 / 4
JSL rel	Jump if less than (signed)	3 / 2	5 / 4
JSLE rel	Jump if less than or equal (signed)	3 / 2	5 / 4
JSG rel	Jump if greater than (signed)	3 / 2	5 / 4
JSGE rel	Jump if greater than or equal (signed)	3 / 2	5 / 4
CJNE A,dir8,rel	Compare directly addressed data to accumulator and jump if not equal	3 / 3	5 / 5
CJNE A,#data,rel	Compare immediate data to accumulator and jump if not equal	3 / 3	5 / 5
CJNE Rn,#data,rel	Compare immediate data to register and jump if not equal	3 / 4	5 / 6
CJNE @Ri,#data,rel	Compare immediate data to ind. and jump if not equal	3 / 4	6 / 7
DJNZ Rn,rel	Decrement register and jump if not zero	2 / 3	5 / 6
DJNZ dir8,rel	Decrement directly addressed location and jump if not zero	3 / 3	6 / 6
TRAP (1)	Jump to the trap interrupt vector	2 / 1	10 / 9
NOP	No operation	1 / 1	1 / 1

Note:
[1] Number of cycle depend on the setting of "uconfig1.4" bit (see Table 54)

When the instruction jumps to another address you should add one state to the number of cycle in the Table 59.

Table 60. Boolean Manipulation

Mnemonic	Description	Bytes	Cycles
CLR C	Clear the carry flag	1 / 1	1 / 1
CLR bit51	Clear directly addressed bit	2 / 2	2 / 2
CLR bit	Clear directly addressed bit	4 / 3	4 / 3
SETB C	Set the carry flag	1 / 1	1 / 1
SETB bit51	Set directly addressed bit	2 / 2	2 / 2
SETB bit	Set directly addressed bit	4 / 3	4 / 3
CPL C	Complement the carry flag	1 / 1	1 / 1
CPL bit51	Complement directly addressed bit	2 / 2	2 / 2
CPL bit	Complement directly addressed bit	4 / 3	4 / 3
ANL C,bit51	AND directly addressed bit to the carry flag	2 / 2	2 / 1
ANL C,bit	AND directly addressed bit to the carry flag	4 / 3	4 / 2
ANL C,/bit51	AND complement of directly addressed bit to the carry flag	2 / 2	2 / 1
ANL C,/bit	AND complement of directly addressed bit to the carry flag	4 / 3	4 / 2
ORL C,bit51	OR directly addressed bit to the carry flag	2 / 2	2 / 1
ORL C,bit	OR directly addressed bit to the carry flag	4 / 3	4 / 2

Mnemonic	Description	Bytes	Cycles
ORL C,/bit51	OR complement of directly addressed bit to the carry flag	2 / 2	2 / 1
ORL C,/bit	OR complement of directly addressed bit to the carry flag	4 / 3	4 / 2
MOV C,bit51	Move directly addressed bit to the carry flag	2 / 2	2 / 1
MOV C,bit	Move directly addressed bit to the carry flag	4 / 3	4 / 2
MOV bit51,C	Move the carry flag to directly addressed bit	2 / 2	2 / 1
MOV bit,C	Move the carry flag to directly addressed bit	4 / 3	4 / 2

3.1.2 Instructions in Hexadecimal Order

Table 61. Instructions in Hexadecimal Order For R80251XC

Opcode Bin.	Opcode Src.	Mnemonic	Opcode Bin.	Opcode Src.	Mnemonic
00 H	**00 H**	NOP	**80 H**	**80 H**	SJMP rel
01 H	**01 H**	AJMP addr11	**81 H**	**81 H**	AJMP addr11
02 H	**02 H**	LJMP addr16	**82 H**	**82 H**	ANL C,bit
03 H	**03 H**	RR A	**83 H**	**83 H**	MOVC A,@A+PC
04 H	**04 H**	INC A	**84 H**	**84 H**	DIV AB
05 H	**05 H**	INC dir8	**85 H**	**85 H**	MOV dir8,dir8
06 H	**A506 H**	INC @R0	**86 H**	**A586 H**	MOV dir8,@R0
07 H	**A507 H**	INC @R1	**87 H**	**A587 H**	MOV dir8,@R1
08 H	**A508 H**	INC R0	**88 H**	**A588 H**	MOV dir8,R0
09 H	**A509 H**	INC R1	**89 H**	**A589 H**	MOV dir8,R1
0A H	**A50A H**	INC R2	**8A H**	**A58A H**	MOV dir8,R2
0B H	**A50B H**	INC R3	**8B H**	**A58B H**	MOV dir8,R3
0C H	**A50C H**	INC R4	**8C H**	**A58C H**	MOV dir8,R4
0D H	**A50D H**	INC R5	**8D H**	**A58D H**	MOV dir8,R5
0E H	**A50E H**	INC R6	**8E H**	**A58E H**	MOV dir8,R6
0F H	**A50F H**	INC R7	**8F H**	**A58F H**	MOV dir8,R7
10 H	**10 H**	JBC bit,rel	**90 H**	**90 H**	MOV DPTR,#data16
11 H	**11 H**	ACALL addr11	**91 H**	**91 H**	ACALL addr11
12 H	**12 H**	LCALL addr16	**92 H**	**92 H**	MOV bit,C
13 H	**13 H**	RRC A	**93 H**	**93 H**	MOVC A,@A+DPTR
14 H	**14 H**	DEC A	**94 H**	**94 H**	SUBB A,#data
15 H	**15 H**	DEC dir8	**95 H**	**95 H**	SUBB A,dir8
16 H	**A516 H**	DEC @R0	**96 H**	**A596 H**	SUBB A,@R0
17 H	**A517 H**	DEC @R1	**97 H**	**A597 H**	SUBB A,@R1
18 H	**A518 H**	DEC R0	**98 H**	**A598 H**	SUBB A,R0
19 H	**A519 H**	DEC R1	**99 H**	**A599 H**	SUBB A,R1
1A H	**A51A H**	DEC R2	**9A H**	**A59A H**	SUBB A,R2
1B H	**A51B H**	DEC R3	**9B H**	**A59B H**	SUBB A,R3
1C H	**A51C H**	DEC R4	**9C H**	**A59C H**	SUBB A,R4
1D H	**A51D H**	DEC R5	**9D H**	**A59D H**	SUBB A,R5

Opcode Bin.	Opcode Src.	Mnemonic	Opcode Bin.	Opcode Src.	Mnemonic
1E H	**A51E H**	DEC R6	**9E H**	**A59E H**	SUBB A,R6
1F H	**A51F H**	DEC R7	**9F H**	**A59F H**	SUBB A,R7
20 H	**20 H**	JB bit,rel	**A0 H**	**A0 H**	ORL C,/bit
21 H	**21 H**	AJMP addr11	**A1 H**	**A1 H**	AJMP addr11
22 H	**22 H**	RET	**A2 H**	**A2 H**	MOV C,bit
23 H	**23 H**	RL A	**A3 H**	**A3 H**	INC DPTR
24 H	**24 H**	ADD A,#data	**A4 H**	**A4 H**	MUL AB
25 H	**25 H**	ADD A,dir8	**A5 H**	**A5 H**	**ESC**
26 H	**A526 H**	ADD A,@R0	**A6 H**	**A5A6 H**	MOV @R0,dir8
27 H	**A527 H**	ADD A,@R1	**A7 H**	**A5A7 H**	MOV @R1,dir8
28 H	**A528 H**	ADD A,R0	**A8 H**	**A5A8 H**	MOV R0,dir8
29 H	**A529 H**	ADD A,R1	**A9 H**	**A5A9 H**	MOV R1,dir8
2A H	**A52A H**	ADD A,R2	**AA H**	**A5AA H**	MOV R2,dir8
2B H	**A52B H**	ADD A,R3	**AB H**	**A5AB H**	MOV R3,dir8
2C H	**A52C H**	ADD A,R4	**AC H**	**A5AC H**	MOV R4,dir8
2D H	**A52D H**	ADD A,R5	**AD H**	**A5AD H**	MOV R5,dir8
2E H	**A52E H**	ADD A,R6	**AE H**	**A5AE H**	MOV R6,dir8
2F H	**A52F H**	ADD A,R7	**AF H**	**A5AF H**	MOV R7,dir8
30 H	**30 H**	JNB bit,rel	**B0 H**	**B0 H**	ANL C,/bit
31 H	**31 H**	ACALL addr11	**B1 H**	**B1 H**	ACALL addr11
32 H	**32 H**	RETI	**B2 H**	**B2 H**	CPL bit
33 H	**33 H**	RLC A	**B3 H**	**B3 H**	CPL C
34 H	**34 H**	ADDC A,#data	**B4 H**	**B4 H**	CJNE A,#data,rel
35 H	**35 H**	ADDC A,dir8	**B5 H**	**B5 H**	CJNE A,dir8,rel
36 H	**A536 H**	ADDC A,@R0	**B6 H**	**A5B6 H**	CJNE @R0,#data,rel
37 H	**A537 H**	ADDC A,@R1	**B7 H**	**A5B7 H**	CJNE @R1,#data,rel
38 H	**A538 H**	ADDC A,R0	**B8 H**	**A5B8 H**	CJNE R0,#data,rel
39 H	**A539 H**	ADDC A,R1	**B9 H**	**A5B9 H**	CJNE R1,#data,rel
3A H	**A53A H**	ADDC A,R2	**BA H**	**A5BA H**	CJNE R2,#data,rel
3B H	**A53B H**	ADDC A,R3	**BB H**	**A5BB H**	CJNE R3,#data,rel
3C H	**A53C H**	ADDC A,R4	**BC H**	**A5BC H**	CJNE R4,#data,rel
3D H	**A53D H**	ADDC A,R5	**BD H**	**A5BD H**	CJNE R5,#data,rel
3E H	**A53E H**	ADDC A,R6	**BE H**	**A5BE H**	CJNE R6,#data,rel
3F H	**A53F H**	ADDC A,R7	**BF H**	**A5BF H**	CJNE R7,#data,rel
40 H	**40 H**	JC rel	**C0 H**	**C0 H**	PUSH dir8
41 H	**41 H**	AJMP addr11	**C1 H**	**C1 H**	AJMP addr11
42 H	**42 H**	ORL dir8,A	**C2 H**	**C2 H**	CLR bit
43 H	**43 H**	ORL dir8,#data	**C3 H**	**C3 H**	CLR C
44 H	**44 H**	ORL A,#data	**C4 H**	**C4 H**	SWAP A
45 H	**45 H**	ORL A,dir8	**C5 H**	**C5 H**	XCH A,dir8
46 H	**A546 H**	ORL A,@R0	**C6 H**	**A5C6 H**	XCH A,@R0
47 H	**A547 H**	ORL A,@R1	**C7 H**	**A5C7 H**	XCH A,@R1

Opcode Bin.	Opcode Src.	Mnemonic	Opcode Bin.	Opcode Src.	Mnemonic
48 H	**A548 H**	ORL A,R0	**C8 H**	**A5C8 H**	XCH A,R0
49 H	**A549 H**	ORL A,R1	**C9 H**	**A5C9 H**	XCH A,R1
4A H	**A54A H**	ORL A,R2	**CA H**	**A5CA H**	XCH A,R2
4B H	**A54B H**	ORL A,R3	**CB H**	**A5CB H**	XCH A,R3
4C H	**A54C H**	ORL A,R4	**CC H**	**A5CC H**	XCH A,R4
4D H	**A54D H**	ORL A,R5	**CD H**	**A5CD H**	XCH A,R5
4E H	**A54E H**	ORL A,R6	**CE H**	**A5CE H**	XCH A,R6
4F H	**A54F H**	ORL A,R7	**CF H**	**A5CF H**	XCH A,R7
50 H	**50 H**	JNC rel	**D0 H**	**D0 H**	POP dir8
51 H	**51 H**	ACALL addr11	**D1 H**	**D1 H**	ACALL addr11
52 H	**52 H**	ANL dir8,A	**D2 H**	**D2 H**	SETB bit
53 H	**53 H**	ANL dir8,#data	**D3 H**	**D3 H**	SETB C
54 H	**54 H**	ANL A,#data	**D4 H**	**D4 H**	DA A
55 H	**55 H**	ANL A,dir8	**D5 H**	**D5 H**	DJNZ dir8,rel
56 H	**A556 H**	ANL A,@R0	**D6 H**	**A5D6 H**	XCHD A,@R0
57 H	**A557 H**	ANL A,@R1	**D7 H**	**A5D7 H**	XCHD A,@R1
58 H	**A558 H**	ANL A,R0	**D8 H**	**A5D8 H**	DJNZ R0,rel
59 H	**A559 H**	ANL A,R1	**D9 H**	**A5D9 H**	DJNZ R1,rel
5A H	**A55A H**	ANL A,R2	**DA H**	**A5DA H**	DJNZ R2,rel
5B H	**A55B H**	ANL A,R3	**DB H**	**A5DB H**	DJNZ R3,rel
5C H	**A55C H**	ANL A,R4	**DC H**	**A5DC H**	DJNZ R4,rel
5D H	**A55D H**	ANL A,R5	**DD H**	**A5DD H**	DJNZ R5,rel
5E H	**A55E H**	ANL A,R6	**DE H**	**A5DE H**	DJNZ R6,rel
5F H	**A55F H**	ANL A,R7	**DF H**	**A5DF H**	DJNZ R7,rel
60 H	**60 H**	JZ rel	**E0 H**	**E0 H**	MOVX A,@DPTR
61 H	**61 H**	AJMP addr11	**E1 H**	**E1 H**	AJMP addr11
62 H	**62 H**	XRL dir8,A	**E2 H**	**E2 H**	MOVX A,@R0
63 H	**63 H**	XRL dir8,#data	**E3 H**	**E3 H**	MOVX A,@R1
64 H	**64 H**	XRL A,#data	**E4 H**	**E4 H**	CLR A
65 H	**65 H**	XRL A,dir8	**E5 H**	**E5 H**	MOV A,dir8
66 H	**A566 H**	XRL A,@R0	**E6 H**	**A5E6 H**	MOV A,@R0
67 H	**A567 H**	XRL A,@R1	**E7 H**	**A5E7 H**	MOV A,@R1
68 H	**A568 H**	XRL A,R0	**E8 H**	**A5E8 H**	MOV A,R0
69 H	**A569 H**	XRL A,R1	**E9 H**	**A5E9 H**	MOV A,R1
6A H	**A56A H**	XRL A,R2	**EA H**	**A5EA H**	MOV A,R2
6B H	**A56B H**	XRL A,R3	**EB H**	**A5EB H**	MOV A,R3
6C H	**A56C H**	XRL A,R4	**EC H**	**A5EC H**	MOV A,R4
6D H	**A56D H**	XRL A,R5	**ED H**	**A5ED H**	MOV A,R5
6E H	**A56E H**	XRL A,R6	**EE H**	**A5EE H**	MOV A,R6
6F H	**A56F H**	XRL A,R7	**EF H**	**A5EF H**	MOV A,R7
70 H	**70 H**	JNZ rel	**F0 H**	**F0 H**	MOVX @DPTR,A
71 H	**71 H**	ACALL addr11	**F1 H**	**F1 H**	ACALL addr11

Opcode Bin.	Opcode Src.	Mnemonic	Opcode Bin.	Opcode Src.	Mnemonic
72 H	**72 H**	ORL C,bit	**F2 H**	**F2 H**	MOVX @R0,A
73 H	**73 H**	JMP @A+DPTR	**F3 H**	**F3 H**	MOVX @R1,A
74 H	**74 H**	MOV A,#data	**F4 H**	**F4 H**	CPL A
75 H	**75 H**	MOV dir8,#data	**F5 H**	**F5 H**	MOV dir8,A
76 H	**A576 H**	MOV @R0,#data	**F6 H**	**A5F6 H**	MOV @R0,A
77 H	**A577 H**	MOV @R1,#data	**F7 H**	**A5F7 H**	MOV @R1,A
78 H	**A578 H**	MOV R0,#data	**F8 H**	**A5F8 H**	MOV R0,A
79 H	**A579 H**	MOV R1,#data	**F9 H**	**A5F9 H**	MOV R1,A
7A H	**A57A H**	MOV R2,#data	**FA H**	**A5FA H**	MOV R2,A
7B H	**A57B H**	MOV R3,#data	**FB H**	**A5FB H**	MOV R3,A
7C H	**A57C H**	MOV R4,#data	**FC H**	**A5FC H**	MOV R4,A
7D H	**A57D H**	MOV R5,#data	**FD H**	**A5FD H**	MOV R5,A
7E H	**A57E H**	MOV R6,#data	**FE H**	**A5FE H**	MOV R6,A
7F H	**A57F H**	MOV R7,#data	**FF H**	**A5FF H**	MOV R7,A

Table 62. Instructions in Hexadecimal Order For MCS-251

Opcode Bin.	Opcode Src.	Mnemonic	Opcode Bin.	Opcode Src.	Mnemonic
A508 H	**08 H**	JSLE rel	**A588 H**	**88 H**	
A509 H	**09 H**	MOV Rm,@Wrj+dis	**A589 H**	**89 H**	LJMP @WRj EJMP @DRk
A50A H	**0A H**	MOVZ Wrj,Rm	**A58A H**	**8A H**	EJMP addr24
A50B H	**0B H**	INC R,#short (1) MOV reg,ind	**A58B H**	**8B H**	
A50C H	**0C H**		**A58C H**	**8C H**	DIV Rm,Rm
A50D H	**0D H**		**A58D H**	**8D H**	DIV WRj,WRj
A50E H	**0E H**	SRA reg	**A58E H**	**8E H**	
A50F H	**0F H**		**A58F H**	**8F H**	
A518 H	**18 H**	JSG rel	**A598 H**	**98 H**	LCALL@WRj ECALL @DRk
A519 H	**19 H**	MOV @WRj+dis,Rm	**A599 H**	**99 H**	ECALL addr24
A51A H	**1A H**	MOVS WRj,Rm	**A59A H**	**9A H**	
A51B H	**1B H**	DEC R,#short (1) MOV ind,reg	**A59B H**	**9B H**	
A51C H	**1C H**		**A59C H**	**9C H**	SUB Rm,Rm
A51D H	**1D H**		**A59D H**	**9D H**	SUB WRj,WRj
A51E H	**1E H**	SRL reg	**A59E H**	**9E H**	SUB reg,op2 (2)
A51F H	**1F H**		**A59F H**	**9F H**	SUB DRk,DRk
A528 H	**28 H**	JLE rel	**A5A8 H**	**A8 H**	
A529 H	**29 H**	MOV Rm,@DRk+dis	**A5A9 H**	**A9 H**	Bit Instructions (3)
A52A H	**2A H**		**A5AA H**	**AA H**	ERET

Opcode Bin.	Opcode Src.	Mnemonic	Opcode Bin.	Opcode Src.	Mnemonic
A52B H	**2B H**		**A5AB H**	**AB H**	
A52C H	**2C H**	ADD Rm,Rm	**A5AC H**	**AC H**	MUL Rm,Rm
A52D H	**2D H**	ADD WRj,WRj	**A5AD H**	**AD H**	MUL WRj,WRj
A52E H	**2E H**	ADD reg,op2 (2)	**A5AE H**	**AE H**	
A52F H	**2F H**	ADD DRk,DRk	**A5AF H**	**AF H**	
A538 H	**38 H**	JG rel	**A5B8 H**	**B8 H**	
A539 H	**39 H**	MOV @DRk+dis,Rm	**A5B9 H**	**B9 H**	TRAP
A53A H	**3A H**		**A5BA H**	**BA H**	
A53B H	**3B H**		**A5BB H**	**BB H**	
A53C H	**3C H**		**A5BC H**	**BC H**	CMP Rm,Rm
A53D H	**3D H**		**A5BD H**	**BD H**	CMP WRj,WRj
A53E H	**3E H**	SLL reg	**A5BE H**	**BE H**	CMP reg,op2 (2)
A53F H	**3F H**		**A5BF H**	**BF H**	CMP DRk,DRk
A548 H	**48 H**	JSL rel	**A5C8 H**	**C8 H**	
A549 H	**49 H**	MOV WRj,@WRj+dis	**A5C9 H**	**C9 H**	
A54A H	**4A H**		**A5CA H**	**CA H**	PUSH op1 (4) MOV DRk,PC
A54B H	**4B H**		**A5CB H**	**CB H**	
A54C H	**4C H**	ORL Rm,Rm	**A5CC H**	**CC H**	
A54D H	**4D H**	ORL WRj,WRj	**A5CD H**	**CD H**	
A54E H	**4E H**	ORL reg,op2 (2)	**A5CE H**	**CE H**	
A54F H	**4F H**		**A5CF H**	**CF H**	
A558 H	**58 H**	JSGE rel	**A5D8 H**	**D8 H**	
A559 H	**59 H**	MOV @WRj+dis,WRj	**A5D9 H**	**D9 H**	
A55A H	**5A H**		**A5DA H**	**DA H**	
A55B H	**5B H**		**A5DB H**	**DB H**	
A55C H	**5C H**	ANL Rm,Rm	**A5DC H**	**DC H**	POP op1 (4)
A55D H	**5D H**	ANL WRj,WRj	**A5DD H**	**DD H**	
A55E H	**5E H**	ANL reg,op2 (2)	**A5DE H**	**DE H**	
A55F H	**5F H**		**A5DF H**	**DF H**	
A568 H	**68 H**	JE rel	**A5E8 H**	**E8 H**	
A569 H	**69 H**	MOV WRj,@DRk+dis	**A5E9 H**	**E9 H**	
A56A H	**6A H**		**A5EA H**	**EA H**	
A56B H	**6B H**		**A5EB H**	**EB H**	
A56C H	**6C H**	XRL Rm,Rm	**A5EC H**	**EC H**	
A56D H	**6D H**	XRL WRj,WRj	**A5ED H**	**ED H**	
A56E H	**6E H**	XRL reg,op2 (2)	**A5EE H**	**EE H**	
A56F H	**6F H**		**A5EF H**	**EF H**	
A578 H	**78 H**	JNE rel	**A5F8 H**	**F8 H**	
A579 H	**79 H**	MOV @DRk+dis,WRj	**A5F9 H**	**F9 H**	
A57A H	**7A H**	MOV op1,reg (5)	**A5FA H**	**FA H**	
A57B H	**7B H**		**A5FB H**	**FB H**	

Opcode Bin.	Opcode Src.	Mnemonic	Opcode Bin.	Opcode Src.	Mnemonic
A57C H	**7C H**	MOV Rm,Rm	**A5FC H**	**FC H**	
A57D H	**7D H**	MOV WRj,WRj	**A5FD H**	**FD H**	
A57E H	**7E H**	MOV reg,op2 (2)	**A5FE H**	**FE H**	
A57F H	**7F H**	MOV DRk,DRk	**A5FF H**	**FF H**	

Note:
1) R = Rm / WRj / DRk
2) op2 are defined in Table 63, Table 64 and Table 65.
3) See Table 66 and Table 67.
4) See Table 68.
5) See Table 70

Table 63. Data Instruction

Instruction	Byte 0		Byte 1		Byte 2		Byte 3
Oper Rmd,Rms	x	C	md	ms			
Oper Wrjd,Wrjs	x	D	jd/2	js/2			
Oper Drkd,Drks	x	F	kd/4	ks/4			
Oper Rm,#data	x	E	m	0000	#data		
Oper Wrj,#data16	x	E	j/2	0100	#data16 (high)		#data16 (low)
Oper Drk,#data16	x	E	k/4	1000	#data16 (high)		#data16 (low)

MOV Drk(h),#data16
MOV Drk,#1data16
CMP Drk,#1data16
	7
	7
	B
		A
		E
		E
			k/4	1100	#data16 (high)		#data16 (low)

Oper Rm,dir8	x	E	m	0001	dir8 addr		
Oper Wrj,dir8	x	E	j/2	0101	dir8 addr		
Oper Drk,dir8	x	E	k/4	1101	dir8 addr		
Oper Rm,dir16	x	E	m	0011	dir16 addr (high)		dir16 addr (low)
Oper Wrj,dir16	x	E	j/2	0111	dir16 addr (high)		dir16 addr (low)
Oper Drk,dir16(1)	x	E	k/4	1111	dir16 addr (high)		dir16 addr (low)
Oper Rm,@Wrj	x	E	j/2	1001	m	00	
Oper Rm,@Drk	x	E	k/4	1011	m	00	

Note:
1) For this instruction, the only valid operation is MOV.

Table 64. High Nibble, Byte 0 Of Data Instruction

x		oper
0010	2	ADD
1001	9	SUB
1011	B	CMP
0100	4	ORL
0101	5	ANL
0110	6	XRL

x		oper
0111	7	MOV

Table 65. Addressing Mode Support For Reg,Op2 Instructions

x	Operation	Rm,#data	Wrj,#data16	Drk,#0data16	Drk,#1data16	Rm,dir8	Wrj,dir8	Drk,dir8	Rm,dir16	Wrj,dir16	Drk,dir16	Rm,@Wrj	Rm,@Drk
2	ADD reg,op2	x	x	x		x	x		x	x		x	x
9	SUB reg,op2	x	x	x		x	x		x	x		x	x
B	CMP reg,op2	x	x	x	x	x	x		x	x		x	x
4	ORL reg,op2	x	x			x	x		x	x		x	x
5	ANL reg,op2	x	x			x	x		x	x		x	x
6	XRL reg,op2	x	x			x	x		x	x		x	x
7	MOV reg,op2	x	x	x	x	x	x	x	x	x	x	x	x

All of the bit instructions in the MCS 251 architecture have opcode A9.

Table 66. Bit Instructions

Instruction	Byte 0		Byte 1			Byte 2	Byte 3
Bit Instr (dir8)	A	9	xxxx	0	bit	dir8 addr	rel addr

Table 67. High Nibble, Byte 1 Of Bit Instructions

xxxx	Bit Instruction
0001	JBC bit
0010	JB bit
0011	JNB bit
0111	ORL Cy,bit
1000	ANL Cy,bit
1001	MOV bit,Cy
1010	MOV Cy,bit
1011	CPL bit
1100	CLR bit
1101	SETB bit
1110	ORL Cy,/bit
1111	ANL Cy,/bit

Table 68. Push/Pop Instructions

Instruction	Byte 0	Byte 1	Byte 2	Byte 3

Instruction	Byte 0		Byte 1		Byte 2	Byte 3
PUSH #data	C	A	md	ms	#data	
PUSH #data16	C	A	jd/2	js/2	#data16 (high)	#data16 (low)
PUSH Rm	C	A	kd/4	ks/4		
PUSH Wrj	C	A	m	0000		
PUSH Drk	C	A	j/2	0100		
MOV Drk,PC	C	A	k/4	1000		
POP Rm	D	A	m	0001		
POP Wrj	D	A	j/2	0101		
POP Drk	D	A	k/4	1101		

Table 69. Control Instructions

Instruction	Byte 0		Byte 1		Byte 2	Byte 3
EJMP addr24	8	A	md	ms	#data	
ECALL addr24	9	A	jd/2	js/2	#data16 (high)	#data16 (low)
LJMP @Wrj	8	9	kd/4	ks/4		
LCALL @Wrj	9	9	m	0000		
EJMP @Drk	8	8	j/2	0100		
ECALL @Drk	9	9	k/4	1000		
ERET	A	A				
JE rel	8	8	rel addr			
JNE rel	7	8	rel addr			
JLE rel	2	8	rel addr			
JG rel	3	8	rel addr			
JSL rel	4	8	rel addr			
JSGE rel	5	8	rel addr			
JSLE rel	0	8	rel addr			
JSG rel	1	8	rel addr			
TRAP	B	9				

Table 70. Displacement/Extended Movs

Instruction	Byte 0		Byte 1		Byte 2	Byte 3
MOV Rm,@WRj+dis	0	9	m	j/2	dis[15:8]	
MOV Wrk,@WRj+dis	4	9	j/2	k2	dis[15:8]	
MOV Rm,@Drk+dis	2	9	m	k/4	dis[15:8]	
MOV WRj,@DRk+dis	6	9	j/2	k/4	dis[15:8]	
MOV @WRj+dis,Rm	1	9	m	j/2	dis[15:8]	
MOV @WRj+dis,WRk	5	9	j/2	k2	dis[15:8]	
MOV @DRk+dis,Rm	3	9	m	k/4	dis[15:8]	
MOV @DRk+dis,WRj	7	9	j/2	k/4	dis[15:8]	
MOVS WRj,Rm	1	A	j/2	m		
MOVZ WRj,Rm	0	A	j/2	m		

Instruction	Byte 0		Byte 1		Byte 2		Byte 3
MOV WRj,@WRj	0	B	j/2	1000	j/2	0000	
MOV WRj,@DRk	0	B	k/4	1010	j/2	0000	
MOV @WRj,WRj	1	B	j/2	1000	j/2	0000	
MOV @DRk,WRj	1	B	k/4	1010	j/2	0000	
MOV dir8,Rm	7	A	m	0001	dir8 addr		
MOV dir8,WRj	7	A	j/2	0101	dir8 addr		
MOV dir8,DRk	7	A	k/4	1101	dir8 addr		
MOV dir16,Rm	7	A	m	0011	dir16 addr (high)		dir16 addr (low)
MOV dir16,WRj	7	A	j/2	0111	dir16 addr (high)		dir16 addr (low)
MOV dir16,DRk	7	A	k/4	1111	dir16 addr (high)		dir16 addr (low)
MOV @WRj,Rm	7	A	j/2	1001	m	0000	
MOV @DRk,Rm	7	A	k/4	1011	m	0000	

Table 71. Inc/Dec Instructions

Instruction	Byte 0		Byte 1		
INC Rm,#short	0	B	m	00	ss
INC Wrj,#short	0	B	j/2	01	ss
INC Drk,#short	0	B	k/4	11	ss
DEC Rm,#short	1	B	m	00	ss
DEC Wrj,#short	1	B	j/2	01	ss
DEC Drk,#short	1	B	k/4	11	ss

Note:

Encoding for ss: ss – short

00 – 1
01 – 2
11 – 4

Table 72. Shifts Instructions

Instruction	Byte 0		Byte 1	
SRA Rm	0	E	m	0000
SRA Wrj	0	E	j/2	0100
SRL Rm	1	E	m	0000
SRL Wrj	1	E	j/2	0100
SLL Rm	3	E	m	0000
SLL Wrj	3	E	j/2	0100

3.1.3 Read-Modify-Write Instructions

Instructions that read a byte from SFR or internal RAM, modify it and rewrite it back, are called "Read-Modify-Write" instructions. When the destination is an I/O port (P0), or a Port bit, these instructions read the output register rather than the pin.

A shaded cell denotes an instruction in the MCS–251 architecture.

Table 73. RMW Instructions

Mnemonic	Description

Mnemonic	Description
ANL dir8,A	AND accumulator to direct
ANL dir8,#data	AND immediate data to direct
ORL dir8,A	OR accumulator to direct
ORL dir8,#data	OR immediate data to direct
XRL dir8,A	Exclusive OR accumulator to direct
XRL dir8,#data	Exclusive OR immediate data to direct
JBC bit51, rel	Jump if bit is set and clear bit
CPL bit51	Complement bit
INC dir8	Increment direct
DEC dir8	Decrement direct
DJNZ dir8,rel	Decrement and jump if not zero
MOV bit51,C	Move the carry flag to direct bit
CLR bit51	Clear bit
SETB bit51	Set bit
JBC bit, rel	Jump if bit is set and clear bit
CLR bit	Clear bit
SETB bit	Set bit
CPL bit	Complement bit
ANL c,bit	AND bit to carry flag
ANL c,/bit	AND bit to carry flag
ORL c,bit	OR bit to carry flag
ORL c,/bit	OR bit to carry flag
MOV bit,c	Move bit to carry flag

3.2. Procedure Calls, Interrupts, Exceptions

Detailed description of the interrupt structure implemented in the R80251XC is provided by specification of ISR subcomponent in section 5.2.

3.3. Device Configuration

The R80251XC provides user design flexibility by configuring certain operating features at device reset. These features fall into the following categories:

- external memory interface (wait state)
- source mode/binary mode opcodes
- selection of bytes stored on the stack by an interrupt

Wait state configurations provide 0, 1, 2, or 3 wait states.

The configuration of the MCS 251 microcontroller is established by the reset routine based on information stored in configuration bytes. The R80251XC microcontroller stores configuration information in two configuration bytes located in code memory.

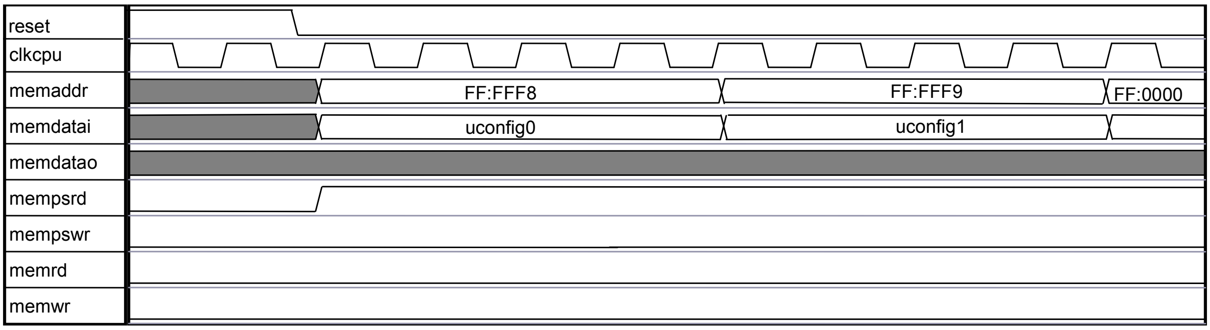


Figure 66. Device Configuration After Reset

The R80251XC reserves the top eight bytes of the memory address map (FF:FFF8H–FF:FFFFH) for an eight-byte configuration array. The two lowest bytes of the configuration array are assigned to the user configuration bytes UCONFIG0 (FF:FFF8H) and UCONFIG1 (FF:FFF9H).

Bit definitions of UCONFIG0 and UCONFIG1 are provided in Table 53 and Table 54. The upper 6 bytes of the configuration array are reserved for future use.

4. Hardware specification

4.1. Block Diagram

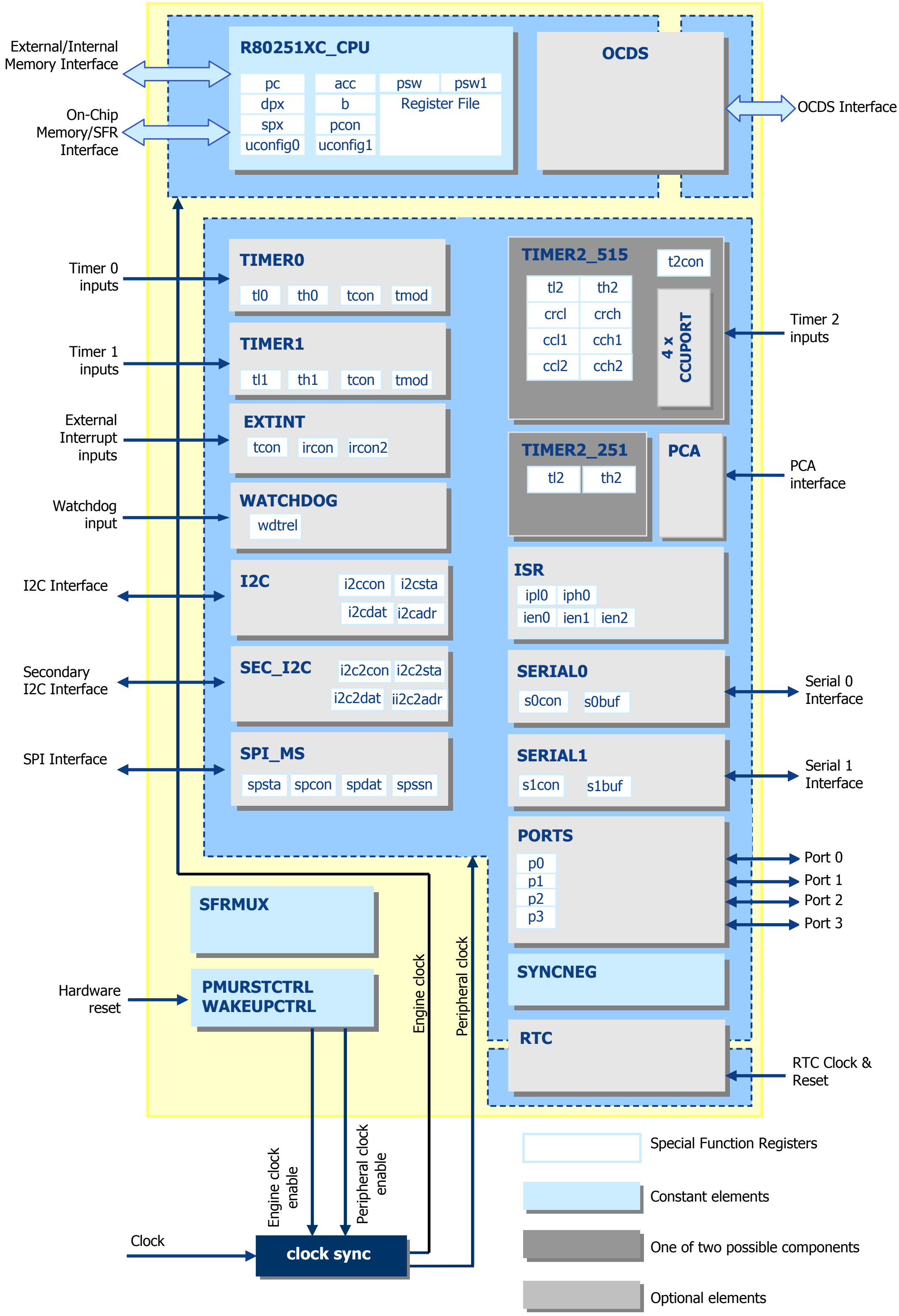


Figure 67. R80251XC Block Diagram

4.2. Blocks Description

The structure of the R80251XC consists of:

- **R80251XC_CPU**– this unit contains the instruction register and instruction execution FSM, Program Counter and Data Pointer register, Stack Pointer register, the Arithmetic-Logic Unit with Accumulator, B and Program Status Word registers (ACC, B, PSW, PSW1) (to provide all arithmetic - addition, subtraction, multiplication, division - and logic - bit- and byte-wise AND, OR, XOR – operations, the Program and External Data Memory interface, the On-Chip Data Memory and Special Function Registers interface, and interface to On-Chip Debug Support
- **ISR** – Interrupt Service Routine, provides interrupt enable and priority registers, priority decoder and interrupt vector generation
- **TIMER0** –provides a flexible 16-bit timer/counter with control and status register - optional
- **TIMER1** –provides a flexible 16-bit timer/counter with control and status register - optional
- **TIMER2_251** – provides a flexible 16-bit timer/counter with control, status and reload/capture register - optional
- **TIMER2_515** – contains the 16-bit Timer 2 with Capture-Compare Unit (4 compare/capture modules), with control and status register - optional
- **PCA** – (Programmable Counter Array) contains the 16-bit Timer/Counter and Capture/Compare Registers (5 modules), with control and status registers - optional
- **SERIAL0** – contains Serial Port 0, a flexible synchronous/UART controller compatible to standard 80C51 serial port, with additional baud rate generator - optional
- **SERIAL1** – provides the Serial Port 1, a simplified UART with its own baud rate generator - optional
- **EXTINT**– provides edge-detection hardware for all External Interrupts from 0 to 12 - optional
- **WATCHDOG** – system supervisor, generating microcontroller reset when not refreshed in specified time - optional
- **DMA** – Direct Memory Access Controller, with 1 to 8 independent channels - optional
- **PMURSTCTRL -** Power Management Unit with Reset Control Unit, generates clock gates for the main CPU and for peripherals, serves the Power Down Modes: IDLE and STOP; generates internal synchronous reset signal (upon external reset or watchdog timer overflow) - optional
- **WAKEUPCTRL –** Wake-Up From Power-Down Mode Control Unit, provides external interrupts “int0” and “int1” service during power-down modes, to be used by the **PMURSTCTRL** module - optional
- **PORTS** – parallel I/O port controller, serves up to 4 parallel 8-bit I/O ports to be used in combination with off-core buffers, compatible to classic 80C51, but without multiplexed memory feature and without alternate functions (if needed, to be combined off-core) - optional
- **SYNCNEG**– contains flip-flops to synchronize all asynchronous inputs such as hardware reset or external interrupts

- **SFRMUX**– provides a common bus bultiplexer for all the internal and external Special Function Registers
- **I2C** – provides a flexible master – slave I2C interface - optional
- **SEC_I2C** – provides secondary master – slave I2C interface - optional
- **SPI** – provides a flexible master – slave SPI interface - optional
- **OCDS** – provides development functions such as run/stop/step control and software/hardware breakpoints of program execution - optional
- **RTC** – provides Real Time Clock function - optional

4.3. Clocks

4.3.1 Clock Inputs

Table 74. R80251XC Clock Inputs

Clock	Type	Polarity	Description
clkcpu	I	Rise	**Engine clock** Pulse for internal circuits, which are stopped when R80251XC is in IDLE or STOP mode
clkper	I	Rise	**Peripheral clock** Pulse for internal circuits, which are stopped when R80251XC is in STOP mode
tck	I	Rise/Fall	**Test (Debug) Clock** IEEE1149.1 port clock input.
rtcx	I	Rise/Fall	**RTC Clock** Pulse for the Real Time Clock (should be 32,768kHz).

- The **"clkcpu"** is a clock signal dedicated to main R80251XC modules like CPU, DMA, OCDS (On Chip Debug System) and SOFTRSTCTRL (Software Reset), which are stopped in both IDLE and STOP power-down modes. It should be generated outside the core the way that it runs when the "clkcpuen" output is active and stopped otherwise.
- The **"clkper"** is a clock signal dedicated to R80251XC peripheral modules like Interrupt Service Routine, Serial PORTS, Timers, Watchdog Timer, Multiplication/Division Unit, Parallel I/O PORTS, External Interrupts, I2C and SPI, which are stopped in the STOP mode. It should be generated outside the core the way that it runs when the "clkperen" output is active and stopped otherwise. Both "clkcpu" and "clkper" have to be balanced since there is no synchronization logic between their domains.
- The **"tck"** is a clock signal dedicated to R80251XC OCDS module. It is used to synchronize the transmission through OCDS port (IEEE1149.1 compliant). Both edges of this clock are used inside the OCDS module.
- The **"rtcx"** is a clock signal dedicated to R80251XC Real Time Clock module. It should be connected to a 32,768kHz oscillator to provide valid time count. The "rtcx" triggers the RTC time registers at the rising edge.

4.3.2 Clock Domains

There are up to four clock domains in the R80251XC.

The first domain, associated with the main system clock – **"clk"** – is composed of:

The first domain, associated with the **"clkcpu"** input, is composed of:
- R80251XC_CPU

The second domain, associated with the **"clkper"** input, consists of:
- ISR
- SERIAL0
- SERIAL1
- TIMER0
- TIMER1
- TIMER2_515 or TIMER_251
- PCA
- EXTINT
- WATCHDOG
- PORTS
- I2C
- SPI_MS
- SYNCNEG

The third domain, associated with the **"tck"** input, consists of:
- OCDS

The fourth domain, associated with the **"rtcx"** input, consists of:
- RTC

4.4. Reset

4.4.1 Reset Description

Upon reset, all the registers and flip-flops of the R80251XC are put into known state. See 2.4.2 for SFR reset values. The Program Memory, External Data Memory, On-Chip Data Memory (Internal Data Memory) and Special Function Register interface control outputs are set inactive. The Program Counter is loaded with zero. During reset, the "swd" input is sampled to enable or disable the "automatic start" of the Watchdog Timer, when selected during core configuration. Only the synchronization flip-flops gathered in the SYNCNEG module have no reset.

4.4.2 Power-On Reset

Power-on reset feature is not implemented in R80251XC.

4.4.3 Hardware Reset

The R80251XC core contains a single "reset" input. It should be active (high) for at least two periods of "clkper" clock to ensure that it will be sampled active at least once.

The "reset" input is routed to the **SYNCNEG** subcomponent and sampled there at every peripheral clock ("clkper") rising edge. The main internal synchronous "rst" signal is generated from the input samples of "reset", software reset, watchdog overflow and OCDS. The "rst" signal is routed to all clock domains in the core and it forces all synchronous logic to a known state.

Detailed description of reset generation can be found in paragraphs describing **SYNCNEG** and **PMURSTCTRL** subcomponent.

Additionally the R80251XC can be reset using the "trst" input to the OCDS module, if implemented. It is used asynchronously to reset the TAP machine of the OCDS, and also generates an internal reset to be OR-ed with the "rst" signal mentioned above.

If the Real Time Clock is implemented, there is the "rtcreset" input implemented to provide asynchronous reset to the flip-flops and registers which are triggered by the "rtcx" clock. Those registers / flip-flops should not be affected by the main hardware reset in order to maintain valid clock count despite of the system activity.

4.4.4 Software Reset

For more details on Software Reset feature see section 5.21. This feature is optional.

4.5. Power Management

4.5.1 Power Saving Modes

There are two power saving modes implemented in the R80251XC. For proper operation it is required that "clkcpu" and "clkper" inputs are off-core connected to "clk" gated by "clkcpuen" and "clkperen" outputs, respectively. The power down modes are controlled by the PCON register (see 2.4.29).

- **STOP mode** - in this mode both **"clkperen"** and **"clkcpuen"** signals are disabled and gated clocks are stopped, causing all synchronous circuits driven by **"clkcpu"** and **"clkper"** to stop.
- **IDLE mode** – in this mode only the main CPU is stopped. That means the Control Unit, ALU, Program/Data Memory interface and RAM/SFR interface are stopped. The **"clkcpuen"** is disabled, while the **"clkperen"** remains active.

4.6. Testability

4.6.1 On-Chip Test and Debug Features

The R80251XC provides the On-Chip Debug Support. This enables the following functions:

- run/stop control
- single-step mode
- software breakpoint
- debugger program
- hardware breakpoints
- access to all registers and memories
- program trace
- data trace

For detailed description of the functions above refer to section OCDS (5.20)

4.6.2 Design For Test

The design is strictly synchronous with positive-edge clocking. There are no internal tri-states. All the flip-flops in the design have synchronous reset (except for SYNCNEG module and RTC). Therefore scan insertion is straightforward.

4.6.3 Asynchronous Inputs

The following input signals should be externally (off-core) synchronized to prevent metastability problems. If synchronized to the system clock, the negative edge should be used.

Table 75. R80251XC Asynchronous Inputs

Name	Type	Polarity	Description
memack	I	High	Memory acknowledge
sfrack	I	High	Special Function Register acknowledge

All signals listed above have combinational logic between core input and internal flip-flops, due to architectural issues coming out of single-clock instruction cycle.

5. Subcomponents specification

5.1. R80251XC_CPU

5.1.1 Overview

The R80251XC_CPU is the main Central Processing Unit of the R80251XC. It is a compact, all-in-one module which integrates the following:

- Shared Program and External Data Memory interface (24-bit linear addressing)
- Interface to internal and external (regarding the whole R80251XC) Special Function Registers
- Internal Memory (up to 1024B of IRAM) interface
- Compact instruction decoder and instruction execution machine
- 8-bit ALU with addition, subtraction, logical operations, bit-wise operations, 8 x 8 and 16 x 16 bit multiplication, 8 / 8 and 16 / 16 bit division
- 24-bit Program Counter (PC)
- 16-bit Stack Pointer (SPX)
- 24-bit Extended Data Pointer (DPX)
- Internal Wait-state generator
- Debug Interface to the OCDS module
- Interface to the PMURSTCTRL module to support the Power-Down modes IDLE and STOP.
- Debug Interface
- Interface to the SFR bus multiplexer for proper functionality of the Read-Modify-Write instructions on the 8-bit I/O ports (PORTS module)
- Interface to the Watchdog Timer (WATCHDOG module) to stop it in debug mode.
- Hold Interface
- DMA Interface

5.1.2 Pin Description

Table 76. R80251XC_CPU Pin Description

Name	Type	Polarity Bus size	Description
Global signals			
clkcpu	I	Rise	**Engine clock** Synchronizes all sequential logic at the rising edge
rst	I	High	**Synchronous reset input** When active for at least 1 clock cycle, the CPU is reset at the rising edge of the "clkcpu"
Program and External Data Memory Bus			
memdatai	I	8	**Memory data input** Shared Program / External Data Memory data input

Name	Type	Polarity Bus size	Description
memaddr	O	24	**Memory address** Shared Program / External Data Memory address output
memdatao	O	8	**Memory data output** Data to be written to Program / External Data Memory
mempsrd	O	High	**Program Memory read enable**
mempswr	O	High	**Program Memory write enable**
memrd	O	High	**External Data Memory read enable**
memwr	O	High	**External Data Memory write enable**
mempsack	I	High	**Program Memory acknowledge**
memack	I	High	**External Data Memory acknowledge**
movc_access	O	High	**Indicates Code Memory read by instruction, not to queue** Used in R80251XC-I(F) version only
codefetch_discard	I	High	**Cancels the actually completed code fetch** Used in R80251XC-I(F) version only
Program and External Data Memory Bus for Rising-Edge Triggered SRAM			
memaddr_comb	O	24	**Memory address**
memdatao_comb	O	8	**Memory data output**
mempsrd_comb	O	High	**Program Memory read enable**
mempswr_comb	O	High	**Program Memory write enable**
memrd_comb	O	High	**External Data Memory read enable**
memwr_comb	O	High	**External Data Memory write enable**
codefetch_comb	O	High	**Denotes Code fetch to queue from code read by MOVC** Used only in the R80251XC-I(F) version
Power Management Unit signals			
cpu_resume	I	High	**Resume input** Resumes the CPU from power-down mode
Interrupt Controller (ISR) signals			
irq	I	High	**Interrupt request input** Forces a call to interrupt service routine pointed by "intvect"
intvect	I	5	**Interrupt vector input** Used to generate the target address of interrupt service routine
intcall	O	High	**Interrupt acknowledge signal** Indicates that the CPU is currently latching the "intvect" and that interrupt is in progress
retiinstr	O	High	**Return from interrupt indicator** Indicates that the CPU is executing the Return From Interrupt instruction, to be used by the Interrupt Controller (ISR)
Other CPU status signals			
newinstr	O	High	**New instruction indicator** Indicates the 1st cycle of a new instruction
rmwinstr	O	High	**Read-Modify-Write instruction indicator** Indicates the CPU is executing a Read-Modify-Write kind of instruction

Name	Type	Polarity Bus size	Description
waitstaten	O	Low	**Waitstate indicator** When low, indicates that the CPU is performing a Wait State
HOLD Mode signals			
hold	I	High	**Hold Request**
holda	O	High	**Hold Acknowledge**
Debug Interface signals			
debugreq	I	High	**Debug Request input** Forces the CPU to stop program execution and enter the Debug Mode
debugprog	I	High	**Debugger Program select input** Determines whether the instruction executed comes from the debugger logic or normal program, and therefore suppresses the incrementation of the Program Counter
debugack	O	High	**Debug Acknowledge output** Indicates that the CPU has stopped instruction execution and entered the Debug Mode
codefetch	O	High	**Instruction Fetch output** Indicates that there is a program memory read to the queue
pcreg	O	24	**Program Counter output**
flush	O	High	**Queue Flush indicator** Active when a jump occurs
flush_comb	O	High	**Queue Flush indicator** Early, pre-register value
lastcycle	O	High	Indicates the last cycle before a new instruction is decoded
swbdreq	O	High	Software Breakpoint detection
pcoutsample	O	High	Indicates when the "pcreg" output is valid (for Trace function)
debuginstr	I	32	Instruction to be executed by setting 'debugstep' and 'debugprog'
a5instr	I	High	Indicates the "prefix" instruction execution
pipefull	O	8	Queue status indicator (each bit represents one level of the queue)
pipeempty	O	4	Returns the number of bytes to be removed from queue in the next cycle, used for Trace and breakpoints
watchdogstop	O	High	Stops the Watchdog when the CPU is in debug mode
dr0_out	O	32	Contents of the DR0 register
Internal (On-Chip) Data Memory & Special Function Registers Interface			
ramdatai	I	8	**Internal (On-Chip) Data Memory Data Bus input** Contains data read from memory
sfrdatai	I	8	**Special Function Registers Data Bus input** Contains data read from Special Function Registers
ramaddr	O	10	**Internal (On-Chip) Data Memory address bus** Shared address bus for Internal (On-Chip) Data Memory
ramaddr_sync	O	10	**Internal (On-Chip) Data Memory address bus** The contents of "ramaddr" after a register, to be used by breakpoints/Trace

Name	Type	Polarity Bus size	Description
sfraddr	O	8	**SFR address bus** Shared address bus for Special Function Registers
sfraddr_comb	O	8	**SFR address bus** Early values (pre-register) of the "sfraddr" to be used in the Interrupt Controller (ISR)
ramdatao	O	8	**Internal (On-Chip) Data Memory bus output** Contains data to be written to Internal (On-Chip) Data Memory
sfrdatao	O	8	**SFR data bus output** Contains data to be written to Special Function Registers
sfrdatao_comb	O	8	**SFR data bus output** Early values (pre-register) of the "sfrdatao" to be used in the Interrupt Controller (ISR)
ramoe	O	High	**Internal (On-Chip) Data Memory Output Enable** Indicates a read access to the memory
ramoe_sync	O	High	The contents of "ramoe" after a register, to be used by breakpoints/Trace
ramwe	O	High	**Internal (On-Chip) Data Memory Write Enable** Indicates a write access to the memory
ramwe_sync	O	High	The contents of "ramwe" after a register, to be used by breakpoints/Trace
sfroe	O	High	**SFR Output Enable** Indicates a read access to the Special Function Register
sfrwe	O	High	**SFR Write Enable** Indicates a write access to the Special Function Register
sfroe_comb	O	High	Early values (pre-register) of the "sfroe" to be used in the Ports module (only in R80251XC-I(F) version)
sfrwe_comb	O	High	Early values (pre-register) of the "sfrwe" to be used in the Interrupt Controller (ISR)
sfrack	I	High	SFR Read / Write Acknowledge
CPU Special Function Registers			
p2	O	High	**Port 2 register** Contains the higher 8 bits of address for MOVX @Ri instructions
acc	O	8	**Accumulator Register** Used by most of the arithmetic and logic instructions as target
b	O	8	**B Register** Used by MUL and DIV instructions, or as a general purpose register
rs_out	O	2	**Register bank select** Select the working register bank of R0-R7 registers
c	O	High	**Carry flag** Carry bit in arithmetic operations and accumulator for Boolean operations

Name	Type	Polarity Bus size	Description
ac	O	High	**Auxiliary Carry flag** Set if there is a carry-out from 3rd bit of Accumulator in BCD operations
ov	O	High	**Overflow flag** Set in case of overflow in Accumulator during arithmetic operations
p	O	High	**Parity flag** Reflects the number of '1's (modulo 2) in the Accumulator
f0	O	High	**General purpose Flag 0** "psw.5" bit from PSW
ud_out	O	High	**User-definable Flag (General purpose Flag 1)** "psw.1" bit from PSW
n_out	O	High	**Negative flag** This bit is set if the result of the last logical or arithmetic operation was negative (i.e., bit 15 = 1). Otherwise it is cleared.
z_out	O	High	**Zero flag** This flag is set if the result of the last logical or arithmetic operation is zero. Otherwise it is cleared.
dpxl_out	O	8	**Data Pointer Extended Low register** DPXL is the lower byte of the upper word of the extended data pointer, DPX = DR56
dph_out	O	8	**Data Pointer High-order Byte register** Contains bits 15:8 of the Data Pointer Register
dpl_out	O	8	**Data Pointer Low-order Byte register** Contains bits 7:0 of the Data Pointer Register
sp_out	O	8	**Stack Pointer Register** Points to the Top-of-stack
sph_out	O	8	**Stack Pointer High Register** SPH is the upper byte of the lower word of DR60, the extended stack pointer (SPX).
gf0	O	-	**General Flag 0** Part of PCON register
gf1	O	-	**General Flag 1** Part of PCON register
uconfig0_7_out	O	-	**Part of UCONFIG0 (bit [7])** For readout, used in R80251XC-I(F) version only
uconfig0_65_out	O	2	**Part of UCONFIG0 (bits [6:5])** For readout
uconfig0_41_out	O	4	**Part of UCONFIG0 (bits [4:1])** For readout, used in R80251XC-I(F) version only
uconfig0_0_out	O	-	**Part of UCONFIG0 (bit [0])** For readout
uconfig1_4_out	O	-	**Part of UCONFIG1 (bit [4])** For readout

Name	Type	Polarity Bus size	Description
uconfig1_3_out	O	-	**Part of UCONFIG1 (bit [3])** For readout, used in R80251XC-I(F) version only
uconfig1_21_out	O	2	**Part of UCONFIG1 (bits [2:1])** For readout
uconfig1_0_out	O	-	**Part of UCONFIG1 (bit [0])** For readout, used in R80251XC-I(F) version only
Power Management Unit signals			
stop	O	High	**Stop Mode indicator** Used to turn-off the CPU and peripheral clocks
idle	O	High	**Idle Mode indicator** Used to turn-off the CPU clock
stop1	O	High	**Stop Mode flag** Used for PCON register readout only
idle1	O	High	**Idle Mode flag** Used for PCON register readout only

5.1.3 Description

The functional blocks of the R80251XC_CPU are designed to provide high configurability, flexibility, readability and performance. This modular structure allows the following parameters to be introduced. The detailed description of functional sub-modules can be found further in this section. Also the exact timing of each instruction execution can be found further.

a) R80251XC_CPU Configuration

The R80251XC_CPU can be tuned-up to the target system requirements by using the following parameters (listed in alphabetical order), to be configured manually or with the use of dedicated proprietary configuration tool.

Table 77. R80251XC_CPU Parameters

Name	Type	Valid values	Default value	Purpose
LEGACY_MODE	INTEGER	0,1	1	Enables/disables the implementation of the "Legacy", Intel 80C251-timing-compatible version (R80251XC-I(F)). When set to 0, the improved version is implemented (R80251XC-T(F)).
HOLD_IMPLEMENT	INTEGER	0,1	1	Enables/disables the implementation of the "hold" and "holda" signals.
OCDS_IMPLEMENT	INTEGER	0,1	1	Enables/disables the implementation of the On-Chip Debug Support
SRST_IMPLEMENT	INTEGER	0,1	1	Enables/disables the implementation of the Software Reset function
PMU_IMPLEMENT	INTEGER	0,1	1	Enables/disables the implementation of Power-Down mode flags of the "pcon" registers, to be used by the PMU module
PORT0_IMPLEMENT PORT1_IMPLEMENT PORT2_IMPLEMENT PORT3_IMPLEMENT	INTEGER	0,1	1	Enable/disable the implementation of the "rmwinstr" output signal which is required when at least one of the parameters is not zero
WATCHDOG_IMPLEMENT	INTEGER	0,1	1	Enables/disables the implementation of the "watchdogstop" output signal used to control the Watchdog Timer in debug mode

b) Program / External Data Memory Interface

The shared Program / External Data Memory interface is built of the following signals:

- "memaddr" 24-bit address bus output
- "memrd" External Memory Read Enable output
- "memwr" External Memory Write Enable output
- "memdatai" 8-bit Data input
- "memdatao" 8-bit Data output

The Program / External Data Memory interface is purely registered. Each of the output signals is connected directly to an output of a flip-flop/register, and the input data bus is captured directly into a register with synchronous write-enable signal.

c) Internal / External SFR Interface

The Special Function Interface consists of the following signals:

- ”sfraddr” 7-bit address bus output
- “sfroe” SFR Read Enable output
- “sfrwe” SFR Write Enable output
- “sfrdatai” 8-bit SFR Data input
- “sfrdatao” 8-bit Data output, shared between SFR and IRAM interfaces

The Internal / External SFR output interface is purely registered. Each of the output signals is connected directly to an output of a flip-flop/register. The input data bus is subject to some combinational logic before it is captured into a register.

d) On-Chip RAM Interface

The On-chip RAM (aka. IRAM) Interface consists of the following signals:

- ”ramaddr” 10-bit address bus output, shared between IRAM and SFR interface
- “ramoe” SFR Read Enable output
- “ramwe” SFR Write Enable output
- “ramdatai” 8-bit SFR Data input
- “ramdatao” 8-bit Data output, shared between IRAM and SFR interfaces

The On-Chip RAM interface is built to support rising-edge triggered SRAM directly. The outputs are combinational early values from before the registers/flip-flops, to be captured by the memory itself. The input data bus is subject to some combinational logic before it is captured into a register.

e) Instruction Decoder and Execution Machine

The Central Processing Unit FSM is the core module of the R80251XC_CPU. Its job is to:

- initiate a program memory read operation (instruction is fetched from queue or from external memory),
- recognize whether there is a Debug Request (“debugreq”), and if so, stop program execution and return the “debugack”,
- recognize whether there is an interrupt request (“irq”), and if so, break current program execution and perform a call to interrupt subroutine,
- decode an instruction and force control signals to ALU, SP, PC, DPTR and Program/XDATA, SFR, or IRAM interfaces; when an instruction requires more than one cycle to complete, the “phase” counter is incremented but the instruction register (top of queue) remains unchanged, and the FSM generates different outputs in different “phases”s; when the last “phase”, which depends on the instruction, is reached, a new instruction is initiated and the process repeats again

The queue fetches successive bytes into internal registers if there is no access to external memory while the instruction is being executed. If there is a data memory read/write, the bus gets busy and instructions are not fetched to the queue. In the last state of an instruction, the queue is shifted by the number of bytes that the instruction consist of. In the last state of Calls, Returns and Jump instructions, all the bytes in the queue are removed and the instruction is fetched from a new address.

The Opcode is placed always in the register pipe[0]. All the operations that can occur within the CPU have their own decoder, which monitors the 'pipe[0]' signal. Together with the "phase" counter, the decoders occasionally enable several signals that direct the data frow between registers and arithmetic operations.

The main FSM also services several signals to the Debug Interface, Interrupt Controller, SFR Multiplexer and Watchdog Timer.

Debug Mode

The Debug Interface inputs are sampled at the beginning of a new instruction, at the same time as the instruction queue gets shifted after completing an instruction. If that sample is active, the FSM proceeds to the Debug Mode: it clears the instruction queue, suppresses program memory fetch, freezes the Program Couter value, and acknowledges using the "debugack" output. In that state, the "debugreq", "debugprog" and "debugstep" inputs are sampled at each clock cycle. The interrupt request is not taken into account in Debug Mode. When the "debugreq" becomes inactive, the FSM resumes program execution by initiating a new instruction fetch and disabling the acknowledge output ("debugack"). When the "debugstep" becomes active during the Debug Mode, the FSM works similary to the situation above but this time it is possible to freeze the state of the Program Counter, when the "debugprog" input was active during active "debugstep". This feature is used to force several instructions from outside to provide access to all resources of the R80251XC, without any influence on program flow. The "debugreq" input must remain active if only a single instruction is to be executed after "debugstep", otherwise the FSM proceeds normal operation as if there is no Debug Mode.

When the "debugreq" input goes active during the Power-Down Mode, it also requests clock resuming to the PMURSTCTRL and when the clock runs, it proceeds to the Debug Mode. The state of the Power-Down flags in the "pcon" register however remain unchanged. The debug interface can execute any instructions from user memory or inserted by debugger, and operate on memories or peripherals as usually. When the "debugreq" is disabled, the CPU returns to the Power-Down Mode as defined by flags in the "pcon" register. Those flags can also be modified by the debug interface.

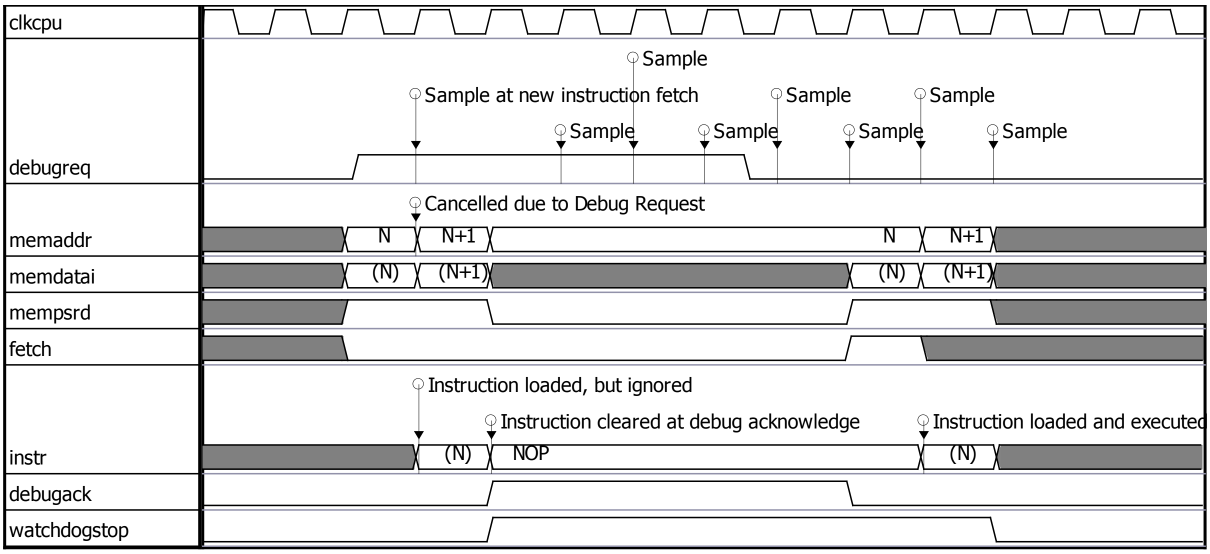


Figure 68. CPU – Debug Request Timing

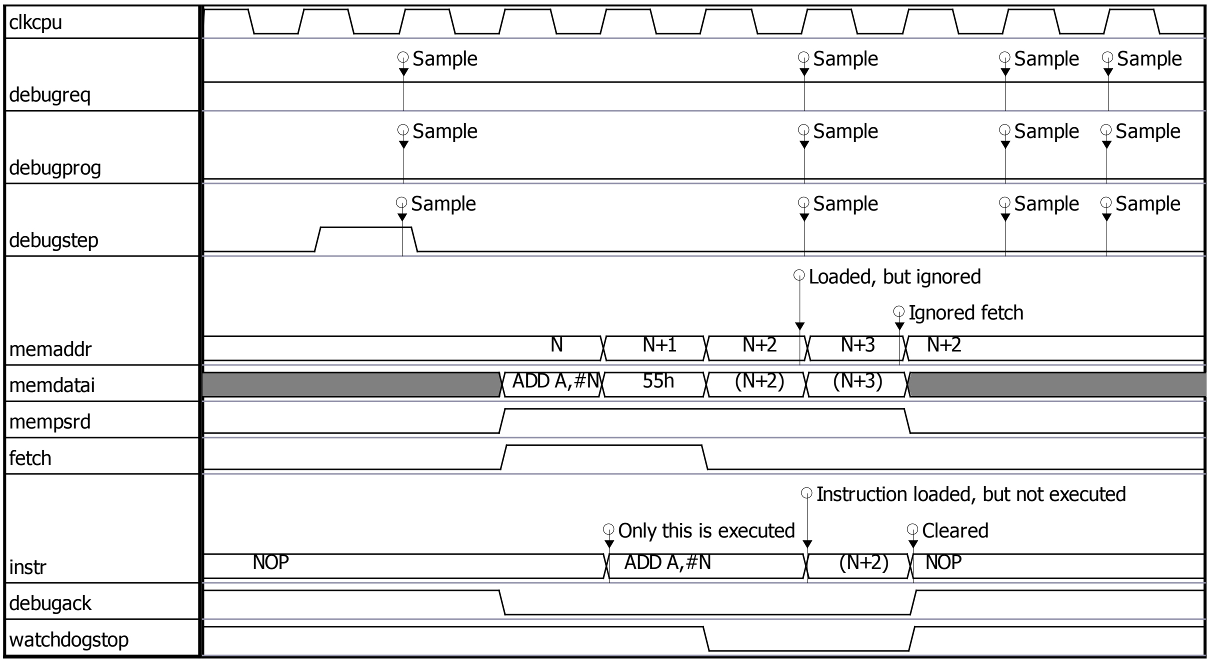


Figure 69. CPU – Single Step in Debug Mode Timing With User Program

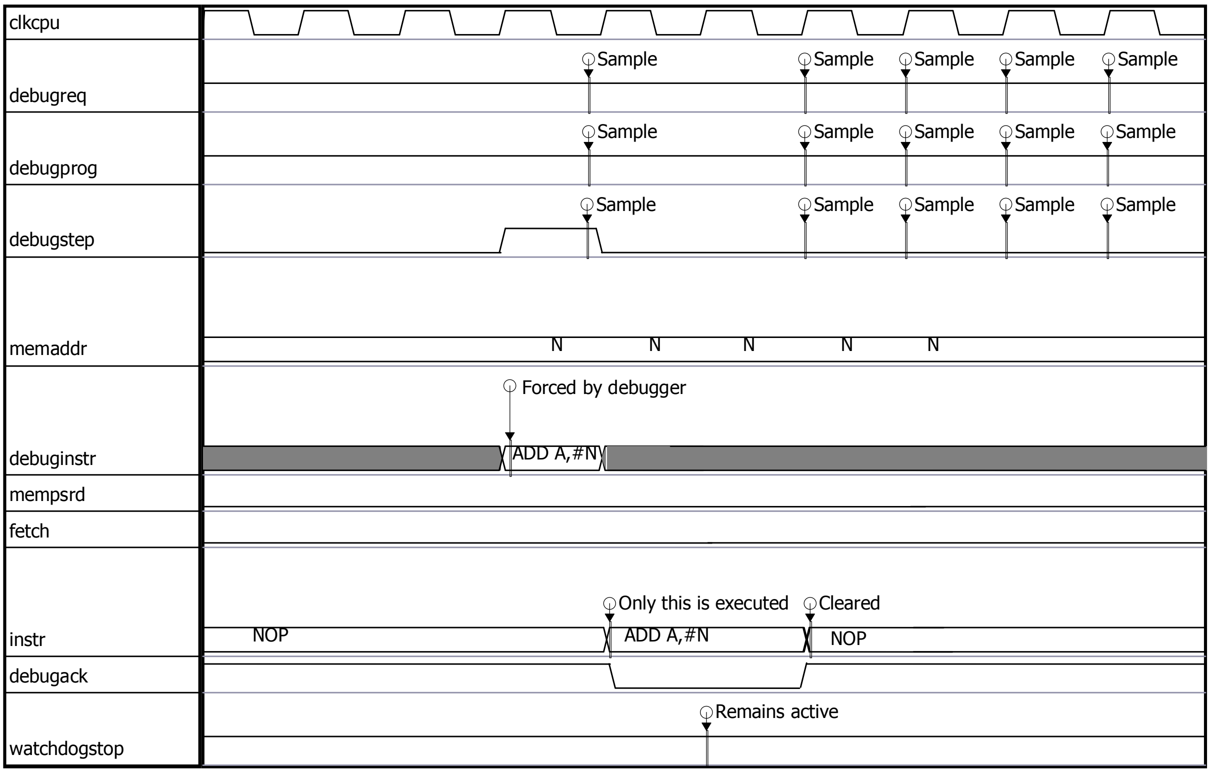


Figure 70. CPU – Single Step in Debug Mode Timing With Debugger Program

Hold Mode

The Hold Interface input is sampled at the beginning of a new instruction, at the same time as the instruction queue is shifted after completion of an instruction. If that sample is active, the FSM proceeds to the Hold Mode: it clears the instruction queue, suppresses program memory fetch, freezes the Program Couter, and acknowledges using the "holda" output. In that state, the "hold" input is sampled at each clock cycle.

The interrupt request is not taken into account in Hold Mode. If during the Hold Mode an interrupt occurs then the core activates "intoccur" output.

When the "hold" becomes inactive, the FSM resumes program execution by initiating a new instruction fetch and disabling the acknowledge output ("holda").

When the "hold" input goes active during the Power-Down Mode it proceeds to the Hold Mode and 'holda' output is activated. The state of the Power-Down flags in the "pcon" register remain unchanged. When the "hold" input is disabled, the CPU deactivates 'holda' output and returns to the Power-Down Mode as defined by flags in the "pcon" register.

If "hold" goes active in Debug Mode and there is no pending single step operation then 'debugack' output is deactivated, the core enters Hold Mode and 'holda' output is activated. When 'hold' input becomes incactive then 'holda' output is deactivated, the core enters Debug Mode and activates 'debugack' output or executes instructions if 'debugreq' input is inactive.

If "hold" goes active in Debug Mode when single-step is executed then step operation is finished, "debugack" stays inactive and "holda" acknowladge output is activated. The core enters Hold Mode. When the "hold" input becomes inactive "holda" output is deactivated, "debugack" goes active and the core enters into Debug Mode.

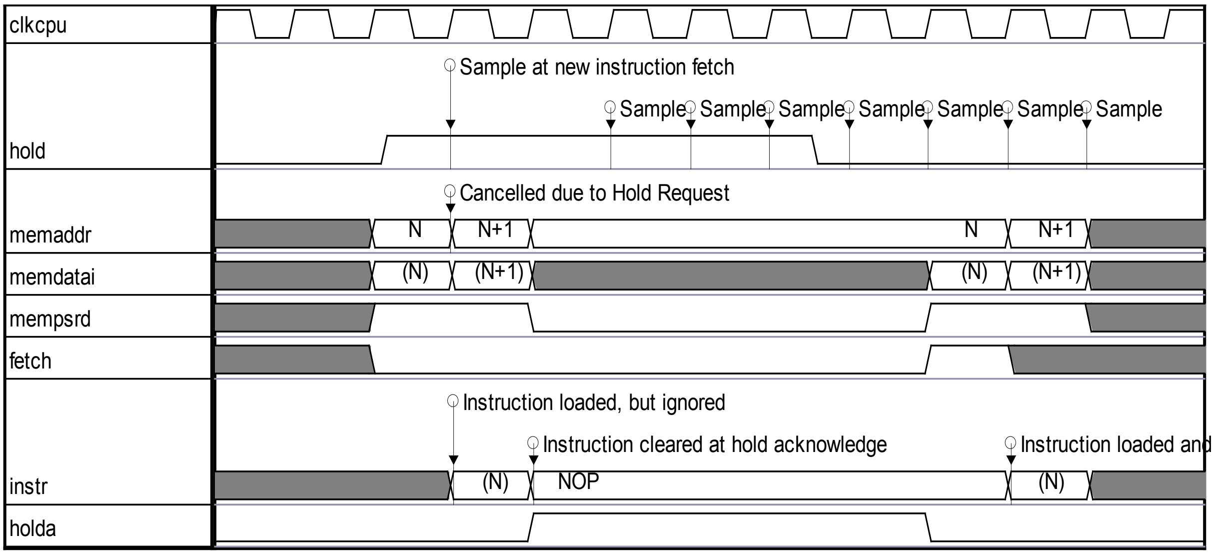


Figure 71. CPU – Hold Request Timing

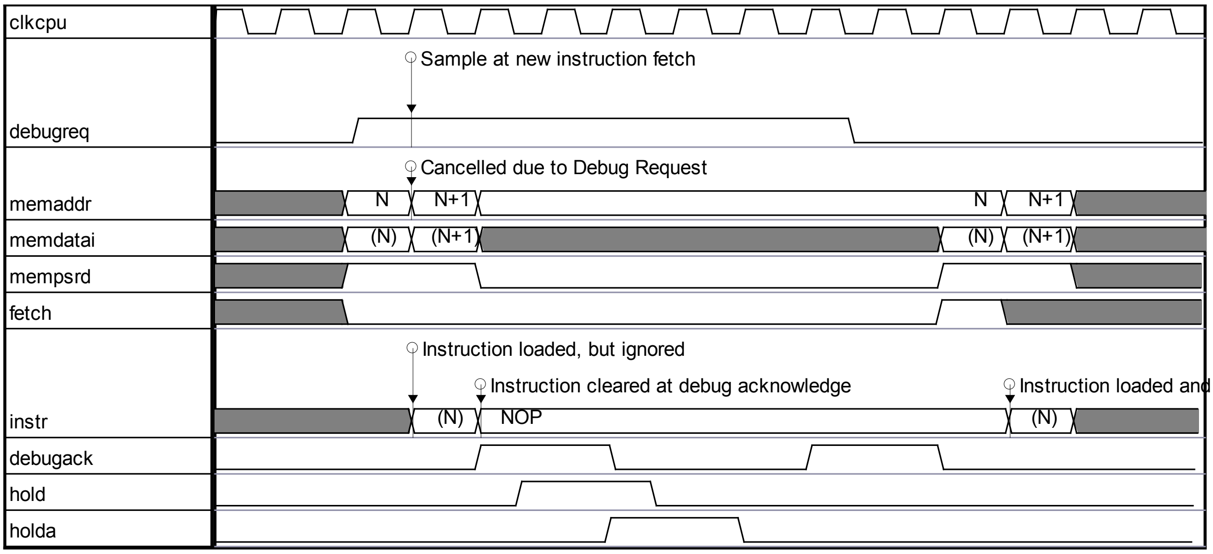


Figure 72. CPU – Hold Request During Debug Mode

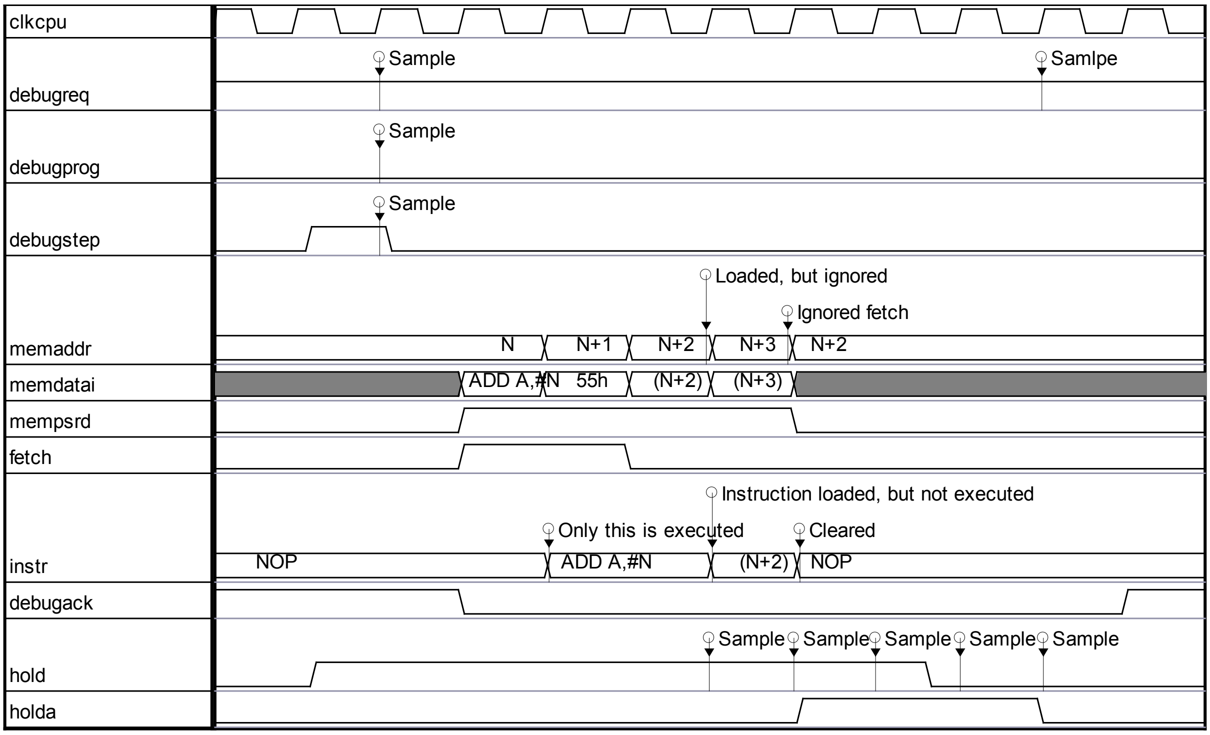


Figure 73. CPU – Hold Request During Single-Step Operation

Interrupts

The interrupt request input “irq” is sampled at the beginning of a new instruction, at the same time as the instruction queue is shifted after completion of an instruction. If that sample is active, and no Debug Mode request sample is active (“debugreq”=0), the the FSM proceeds the interrupt call: suppresses program memory fetch, pushes the Program Counter onto stack, loads the Program Counter with target interrupt vector, acknowledges using the “intcall” output for one clock period, and invokes back the instruction fetch. The actual interrupt vector is sampled at input “intvect” in exactly the cycle when “intcall” output is active, so that the Interrupt Controller (ISR module) can determine which vector was captured. The ISR should then remove the “irq” request immediately as it receives the “intcall”, to prevent multiple interrupt call from single request.

When there was an instruction modifying one of the Interrupt Enable / Priority SFRs, the next following instruction is not interrupted. Therefore if there is a series of instructions writing to e.g. The “ien0” register, an interrupt can be acknowledged after the next instruction after that series.

The other signal which is used by the ISR is the “retiinstr”. It goes active when the RETI (Return from Interrupt) instruction is executed by the FSM. It tells the ISR that an interrupt service subroutine has completed so that it can free its “interrupt in-service” register.

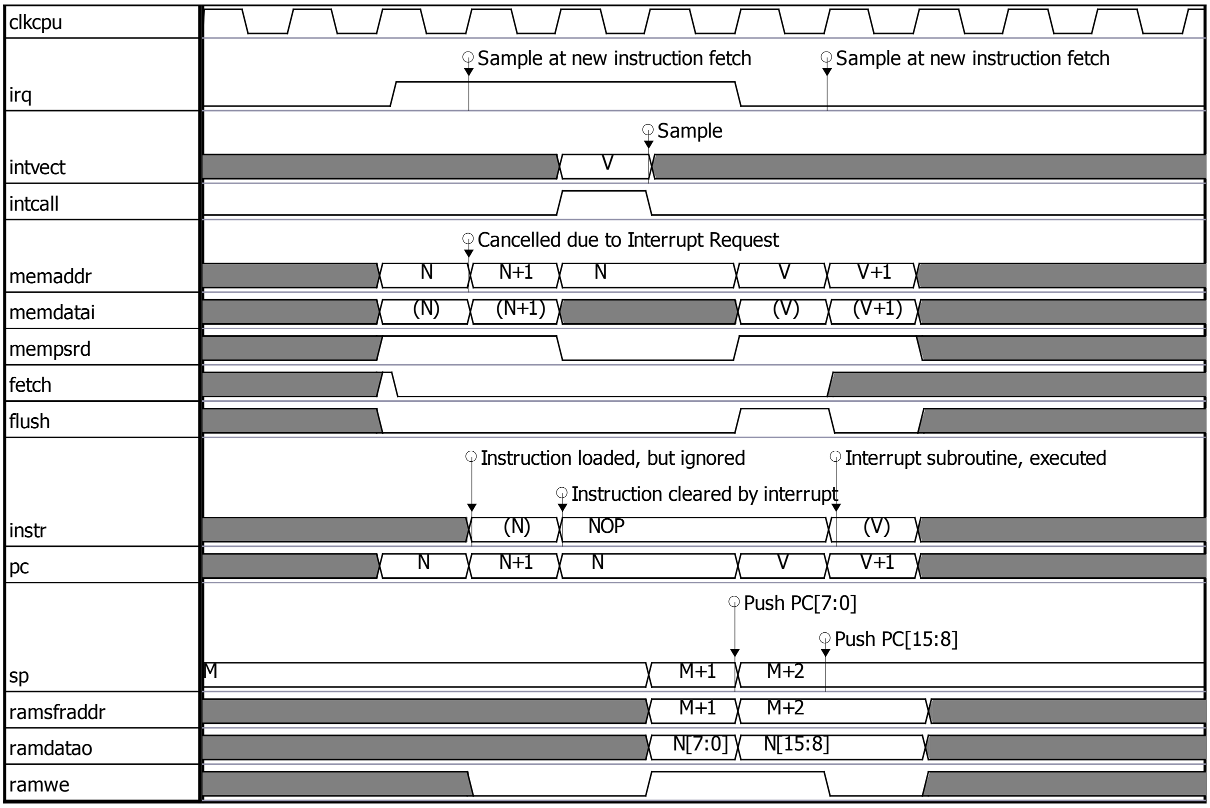


Figure 74. CPU - Interrupt Timing

I/O PORTS SFR control

The “rmwinstr” output is used by the SFRMUX module to select between the output register and port input when referring to “port0”...”port3” SFR for read. Since the so called “Read-Modify-Write” (RMW) instructions read data from output port register rather than port input as other instructions do, the Special Function Register multiplexer uses the “rmwinstr” signal to choose the proper register for read.

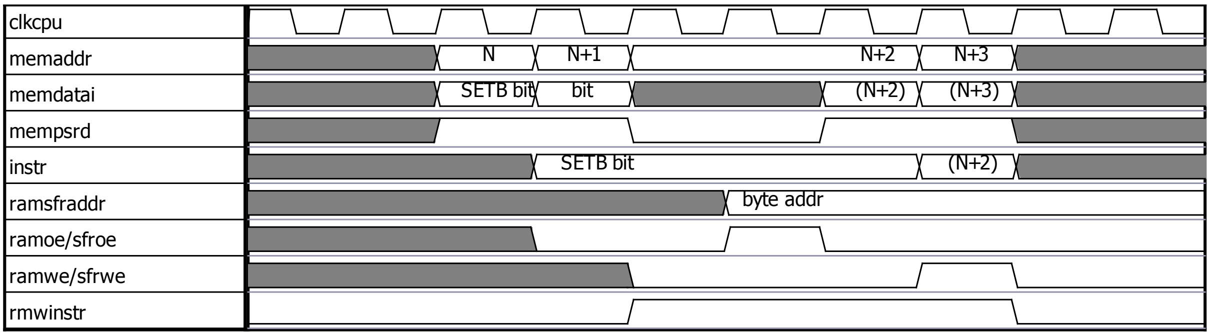


Figure 75. CPU – "Rmwinstr" Output Timing

f) Alu

The Arithmetic-Logic Unit integrated within the R80251XC_CPU consists of the following blocks designed to perform different kinds of operations on 8-bit, 16-bit and 32-bit data:

- basic arithmetic/logic operations
  - addition with or without carry
  - subtraction with or without borrow
  - incrementation / decrementation
  - logical AND, OR, XOR, negation
  - rotate left / right through carry or without carry
  - nibble-swap / nibble write / pass-through / zero
- bit-wise logic operations
  - set / clear / complement / move bit
  - AND / OR bit by carry / not carry
- advanced arithmetic operations
  - decimal adjust
  - multiplication
  - division

The ALU serves arithmetic/logic operations desired by particular instructions, both explicit and implicit. The explicit are e.g. addition in ADD instructions, the implicit are e.g. pass-through operations in MOV instructions, subtractions used for comparison in CJNE instructions.

g) Program Counter

The 24-bit Program Counter is used to generate the Program Memory address during instruction fetch operation. Its contents is copied to the Program / External Data Memory interface and then incremented.

h) Extended Stack Pointer

The 16-bit Extended Stack Pointer is an internal SFR used to address the On-chip Memory (IRAM) or External Ram when operating as a software stack. When the CPU is executing a PUSH or CALL or interrupt service, it forces the Stack Pointer to increment. When executing a POP or RET, the Stack Pointer gets decremented.

i) Extended Data Pointer

While instructions in the MCS 51 architecture always use DPTR as the data pointer, instructions in the MCS 251 architecture can use any word or dword register as a data pointer. DPXL, the byte in location 57, specifies the region of memory (00:–FF:) that maps into the 64-Kbyte external data memory space in the MCS 51 architecture. in other words, the MOVX instruction addresses the region specified by DPXL when it moves data to and from external memory.
Instructions in the MCS® 51 architecture use DPTR for data moves, code moves, and for a jump instruction (JMP @A+DPTR).

j) PMU Interface

The R80251XC_CPU contains the "idle" and "stop" flags of the "pcon" register (see 2.4.29). They are implemented depending on the IP core configuration, see Table 77. When any of these is set using the SFR interface, the "cpu_stop" signal is activated to suppress new instruction fetch and execution. The main FSM of the R80251XC_CPU enters the state when fetch is not performed, the instruction queue is cleared, and all bus control signals (write / read strobes) are inactive. After that, the actual "idle" or "stop" output of the R80251XC_CPU is activated, so that the PMU module can safely turn off the CPU clock ("clkcpu"). to resume the R80251XC_CPU operation, the PMU activates the "cpu_resume" signal which clears the "fsm_halt" mentioned above and the CPU starts instruction fetch again.

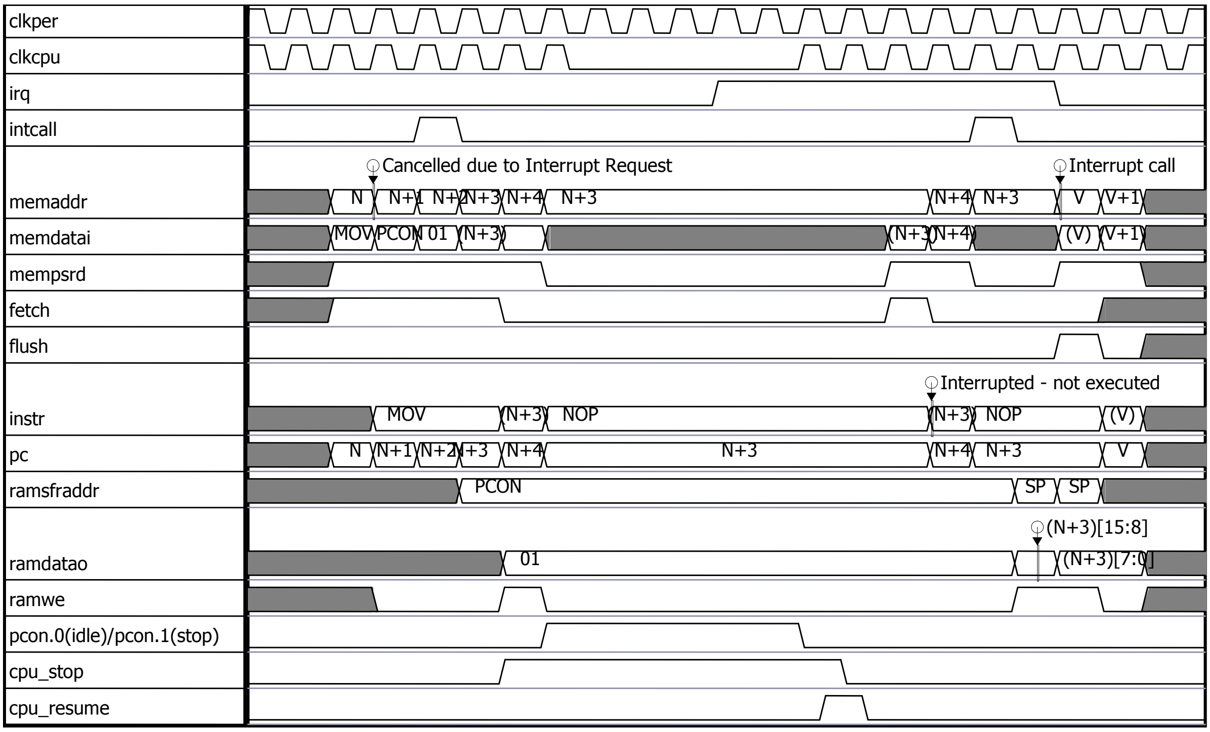


Figure 76. CPU – Power-Down Mode Timings

5.2. ISR

5.2.1 ISR Overview

The **ISR** - Interrupt Service Routine unit, is a subcomponent responsible for interrupt handling. It receives up to 21 interrupt requests. Each interrupt source has its own request flag that is located in devices which is a source of interrupt. No interrupt request flags are located directly in ISR. All interrupts are requested by high level on correspondent inputs to ISR.

Each of the interrupt sources can be individually enabled or disabled by corresponding enable flag in "ien0", "ien1", "ien2" and "ien4" SFR register. Additionally all interrupts can be globally enabled or disabled by the "ea" flag in the "ien0" SFR.

All interrupt sources are divided into 7 interrupts groups. Each of the interrupt groups can have one of four interrupt priority levels assigned. The interrupt priority level is defined by flags located in the "ip0" and "ip1" SFR registers.

5.2.2 ISR Pin Description

Table 78. ISR Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Clock input** Triggers all internal synchronous elements at the rising edge
rst	I	High	**Synchronous reset input** When active for at least 1 clock cycle, the ISR is reset at the rising edge of the "clkper"
intcall	I	High	**Interrupt Call** Signal set by CPU. It is high for one clock cycle when CPU is in interrupt acknowledge cycle, and when it reads interrupt vector.
retiinstr	I	High	**RETI request** Signal set by CPU when it executes RETI instruction
isr_tm	I	High	**ISR Test Mode input** When set to 1, the interrupt vectors assigned to Timer 0 & 1, Serial Port 0 & 1, SPI and I2C interfaces can be triggered only with the use of external inputs of the core (for verification purposes)
int_vect_03	I	High	**Interrupt request 0** Interrupt request input. When it is high, it indicates that some device needs service. If this interrupt is enabled then request to the CPU and the corresponding vector are generated. to assure that interrupt will be accepted by CPU it must be kept high as long as corresponding interrupt acknowledge signal is generated. The following "int_vect_xx" inputs represent the other 21 interrupt sources, where the last part of their name indicates the actial interrupt vector address.
int_vect_0B	I	High	**Interrupt request 1**
int_vect_13	I	High	**Interrupt request 2**
int_vect_1B	I	High	**Interrupt request 3**
int_vect_23	I	High	**Interrupt request 4**

Name	Type	Polarity Bus size	Description
int_vect_2B	I	High	**Interrupt request 5**
int_vect_33	I	High	**Interrupt request 18**
int_vect_43	I	High	**Interrupt request 6**
int_vect_4B	I	High	**Interrupt request 7**
int_vect_53	I	High	**Interrupt request 8**
int_vect_5B	I	High	**Interrupt request 9**
int_vect_63	I	High	**Interrupt request 10**
int_vect_6B	I	High	**Interrupt request 11**
int_vect_EB	I	High	**Interrupt request 29**
int_vect_83	I	High	**Interrupt request 12**
int_vect_8B	I	High	**Interrupt request 13**
int_vect_93	I	High	**Interrupt request 14**
int_vect_9B	I	High	**Interrupt request 15**
int_vect_A3	I	High	**Interrupt request 16**
int_vect_AB	I	High	**Interrupt request 17**
int_vect_BB	I	High	**Interrupt request 23**
irq	O	High	**Interrupt request output** Interrupt request to the CPU. When it is active, it indicates that one or more devices need service.
intvect	O	5	**Interrupt Vector** Interrupt vector is an 5-bit value that CPU uses to generate the address of interrupt service subroutine. The CPU reads this value when intcall signal is set to high.
int_ack_03	O	High	**Interrupt acknowledge 0 output** Interrupt acknowledge signal. It is generated for one clock cycle. When high indicates that CPU has accepted interrupt request and now it calls to corresponding interrupt service subroutine.
int_ack_0B	O	High	**Interrupt acknowledge 1**
int_ack_13	O	High	**Interrupt acknowledge 2**
int_ack_1B	O	High	**Interrupt acknowledge 3**
int_ack_43	O	High	**Interrupt acknowledge 6**
int_ack_4B	O	High	**Interrupt acknowledge 7**
int_ack_53	O	High	**Interrupt acknowledge 8**
int_ack_5B	O	High	**Interrupt acknowledge 9**
int_ack_63	O	High	**Interrupt acknowledge 10**
int_ack_6B	O	High	**Interrupt acknowledge 11**
int_ack_8B	O	High	**Interrupt acknowledge 13**
int_ack_93	O	High	**Interrupt acknowledge 14**
int_ack_9B	O	High	**Interrupt acknowledge 15**
int_ack_A3	O	High	**Interrupt acknowledge 16**
int_ack_AB	O	High	**Interrupt acknowledge 17**
int_ack_BB	O	High	**Interrupt acknowledge 23**

Name	Type	Polarity Bus size	Description
is_reg	O	4	**Interrupt in Service register** Returns the priority level of the interrupt actually being in service. Used to generate an interrupt in the power-down mode
ip0	O	7	**Interrupt Priority register 0 (IPL0)** Output of Interrupt Priority 0 SFR register. Only the bits used for interrupt handling purpose are implemented in the ISR (7 bits).
ip1	O	7	**Interrupt Priority register 1 (IPH0)** Output of Interrupt Priority 1 SFR register. Only the bits used for interrupt handling purpose are implemented in the ISR (7 bits).
ien0	O	7	**Interrupt Enable register 0** Interrupt Enable 0 SFR register output.
ien1	O	6	**Interrupt Enable register 1** Interrupt Enable 1 SFR register output.
ien2	O	6	**Interrupt Enable register 2** Interrupt Enable 2 SFR register output.
ien4	O	6	**Interrupt Enable register 4** Interrupt Enable 4 SFR register output.
sfraddr	I	7	**SFR register Address bus** Address of SFR register that CPU wants to read from or write to.
sfrdatai	I	8	**SFR input data bus** Data that CPU wants to write to SFR register. Destination register is selected by address on sfraddr bus.
sfrwe	I	High	**SFR write enable** Write enable signal to SFR registers. When it is active (high) the CPU writes data (from sfrdatai) to register selected by address on the "sfraddr".

5.2.3 ISR Block Diagram

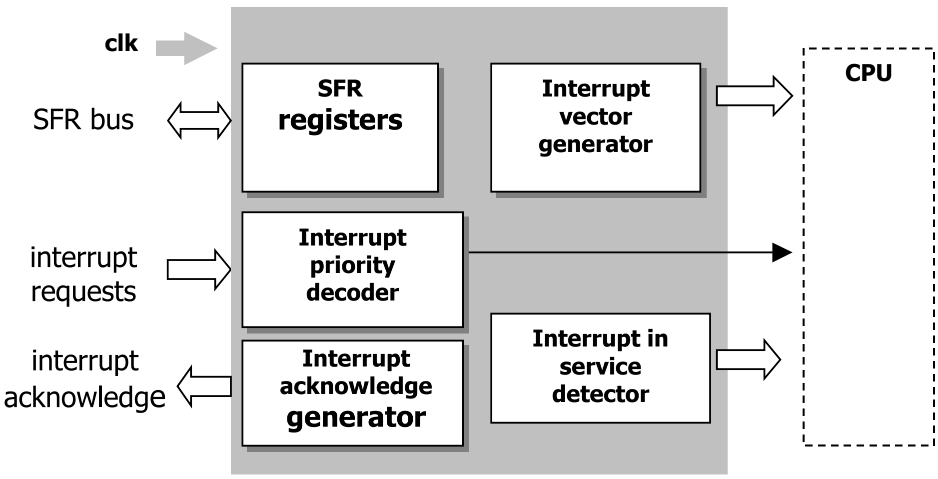


Figure 77. ISR Block Diagram

5.2.4 ISR Description

a) ISR Special Function Registers Block

The following SFR registers are located in the ISR:

- ien0 ien1 ien2 ien4 – interrupt enable registers
- ipl0 iph0 – interrupt priority registers.

There are no interrupt request flags inside the ISR unit.

More detailed description of the Special Function Registers can be found in section 2.3.4 .

Writing to the SFR registers located in the ISR unit is done through SFR bus. When the CPU wants to write data to SFR registers located in the SFR unit, it sets corresponding address on the SFR address bus ("sfraddr" signal) and activates the "sfrwe" signal for one clock cycle. When the "sfrwe" is active (high), data from "sfrdatai" bus are written into the SFR register pointed by address on "sfraddr" bus.

Outputs from all SFR registers are mapped to the output ports of the ISR. The multiplexer that chooses one of them when the CPU reads the SFR register is outside of the ISR unit.

b) Interrupt Priority Decoder

All interrupt requests are divided into 7 groups. The definition each of group is shown in table below.

Table 79. Interrupt Priority Groups

Interrupt group	Priority in group:		
	Highest	Middle	Lowest
Group0	int_vect_03	int_vect_83	int_vect_43
Group1	int_vect_0B	int_vect_8B	int_vect_4B
Group2	int_vect_13	int_vect_93	int_vect_53
Group3	int_vect_1B	int_vect_9B	int_vect_5B
Group4	int_vect_23	int_vect_A3	int_vect_63
Group5	int_vect_2B	int_vect_AB	int_vect_6B
Group6	int_vect_33	int_vect_BB	int_vect_EB

Inside group there is, fixed by hardware, interrupt priority structure. Sources from first column have the highest priority (inside group), sources from middle column have middle priority and interrupts sources from the last column have the lowest priority. There is no possibility to change interrupt priority inside the group.

There is also an interrupt priority structure between the groups. Group0 has the highest priority and Group5 has the lowest. The priority between groups can be programmed by changing priority level (priority level can be set from 0 to 3) assigned to each group. The priority level of interrupt group is defined by flags of the "ipl0" and "iph0" SFR registers. Relations between flags in "ipl0" and "iph0" SFR registers and interrupt groups are defined in the table below.

Table 80. Interrupt priority between groups

Interrupt group	Priority within this same priority level	IPH0 bit	IPL0 bit

Interrupt group	Priority within this same priority level	IPH0 bit	IPL0 bit
Group0	Highest	0	0
Group1		1	1
Group2		2	2
Group3		3	3
Group4		4	4
Group5		5	5
Group6	Lowest	6	6

The settings of interrupt priority flags and priority level are shown in table below.

Table 81. Interrupt Priority Levels

Level	Priority	IPH0.x bit	IPL0.x bit
Level0	Lowest	0	0
Level1		0	1
Level2		1	0
Level3	Highest	1	1

All priority types are taken into account when more than one interrupt is requested. The most important is the priority level set by "ipl0" and "iph0" registers, then the natural priority between groups, and at last the priority inside each group.

To determine which interrupt has the highest priority (which must be serviced in the first order) the following steps are made.

1. From all groups there are chosen those which have the highest priority level.
2. From those with the highest priority level there is chosen one with the highest natural priority between groups.
3. From group with highest priority there is chosen the interrupt with the highest priority inside the group.

What is important is that currently running interrupt service subroutine can be interrupted only by interrupt with higher priority level. No interrupt with the same or lower priority level can interrupt the currently running interrupt service subroutine. Therefore there can be maximum four interrupts in service at the same time.

The information about priority level of the interrupt in service is stored in the "is_reg" register. When any of the bits in the "is_reg" is set, that means that interrupt with corresponding priority level is in service. Table 82 shows the relation between bits in "is_reg" register and interrupts in service.

Table 82. The "IS_REG" Register Bits Function

is_reg bit	Interrupt in service
is_reg.0	Priority level 0
is_reg.1	Priority level 1
is_reg.2	Priority level 2
is_reg.3	Priority level 3

ATTENTION: Any write to "ipl0", "iph0" SFRs registers (changing interrupt priority levels) does not modify the "is_reg" register. That means that when some interrupt is requested with priority X, the invoked subroutine has priority level X even though priority level for sources has been changed; this interrupt subroutine can be interrupted only by interrupt with priority higher than X.

The interrupt priority structure is built as a combinational block. It is split into three parts. Figure 78 shows structure of interrupt priority hardware.

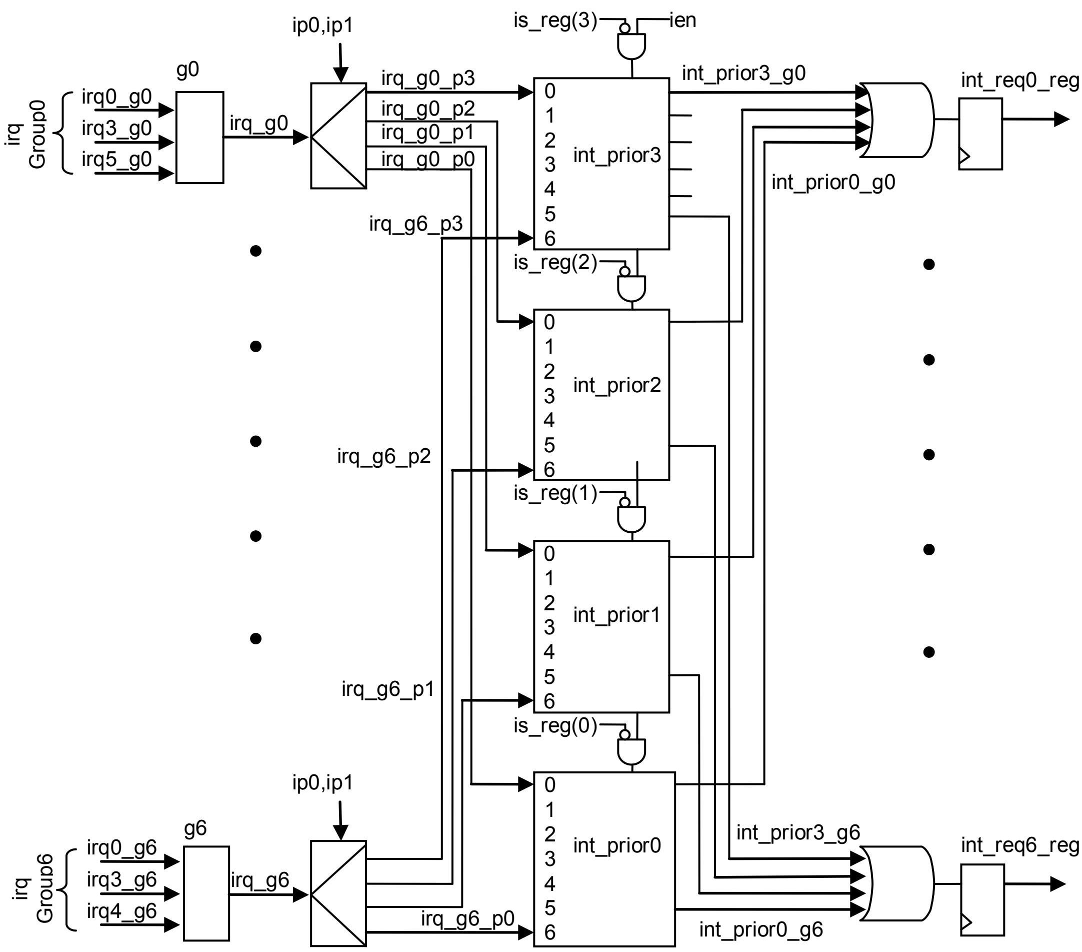


Figure 78. Priority Structure Diagram

In the first step, all interrupt request inputs are masked by logical AND with the corresponding interrupt enable bits. After the masking logic, interrupt requests are divided into 6 groups. The relation between interrupt request inputs, interrupt enable bits and interrupt grouping (as in diagram above) is shown in Table 83.

Table 83. Interrupt Request Relations

Interrupt request input	Interrupt enable bit	Interrupt request in diagram	Group	Interrupt
int_vect_03	ien0(0)	irq0_g0	Group 0	irq_g0(4)
int_vect_83	ien2(0)	irq2_g0		irq_g0(2)
int_vect_43	ien1(0)	irq4_g0		irq_g0(0)

Interrupt request input	Interrupt enable bit	Interrupt request in diagram	Group	Interrupt
int_vect_0B	ien0(1)	irq0_g1	Group 1	irq_g1(4)
int_vect_8B	ien2(1)	irq2_g1		irq_g1(2)
int_vect_4B	ien1(1)	irq4_g1		irq_g1(0)
int_vect_13	ien0(2)	irq0_g2	Group 2	irq_g2(4)
int_vect_93	ien2(2)	irq2_g2		irq_g2(2)
int_vect_53	ien1(2)	irq4_g2		irq_g2(0)
int_vect_1B	ien0(3)	irq0_g3	Group 3	irq_g3(4)
int_vect_9B	ien2(3)	irq2_g3		irq_g3(2)
int_vect_5B	ien1(3)	irq4_g3		irq_g3(0)
int_vect_23	ien0(4)	irq0_g4	Group 4	irq_g4(4)
int_vect_A3	ien2(4)	irq2_g4		irq_g4(2)
int_vect_63	ien1(4)	irq4_g4		irq_g4(0)
int_vect_2B	ien0(5)	irq0_g5	Group 5	irq_g5(4)
int_vect_AB	ien2(5)	irq2_g5		irq_g5(2)
int_vect_6B	ien1(5)	irq4_g5		irq_g5(0)
int_vect_33	ien0(6)	irq0_g6	Group 6	irq_g6(4)
int_vect_BB	ien2(6)	irq2_g6		irq_g6(2)
int_vect_EB	ien1(6)	irq4_g6		irq_g6(0)

The first block in interrupt priority structure (shown as "g0" to "g6" in diagram above) chooses which interrupt inside the group has the highest priority. The output from this block is a 3-bit vector that has only one bit active. This active bit represents interrupt request with the highest priority inside the group. in case when no interrupt has been requested all outputs are low ('0'). The truth table for this component is shown in Table 84.

Table 84. Group 0 Requests Priority Truth Table

irq0_g0	irq1_g0	irq2_g0	irq_g0
0	0	0	00000
0	0	1	00001
0	1	0	00100
0	1	1	00100
1	0	0	10000
1	0	1	10000
1	1	0	10000
1	1	1	10000

The table above is made for Group 0. For other groups the relations are built in this same way. The only exception is Group 6, which has two interrupt source only.

The next component in interrupt priority structure is the demultiplexer that assigns interrupt requests to one of four interrupt priority level blocks. Flags in "ipl0" and "iph0" SFR registers define to which priority level block the current interrupt request should be assigned. Outputs of this demultiplexer that are not pointed by "ipl0.x" and "iph0.x" bits are set to inactive state ('0').

Table 85. Group 0 Demultiplexer Truth Table

input	ip1.0, ip0.0	irq_g0_p3	irq_g0_p2	irq_g0_p1	irq_g0_p0
irq_g0	00	00000	00000	00000	irq_g0
	01	00000	00000	irq_g0	00000
	10	00000	irq_g0	00000	00000
	11	irq_g0	00000	00000	00000

Table 85 describes the behavior of priority level demultiplexer for Group0. The other groups are built in the same way.

The next elements in interrupt priority structure are 4 priority decoders (blocks "int_prior0" to "int_prior3") that represent four priority levels. Block "int_prior0" represents interrupt priority level 0, "int_prior1" represents level 1 and so on. Inputs to this block are outputs from demultiplexers that assign groups of interrupt requests to priority levels exactly like bits in "ip0" and "ip1" SFR define. Additionally each priority level block has an enable input ("enable_in"). If this pin is low, all interrupt requests to this level are ignored. Output from level decoder block is one interrupt request which has the highest priority inside the level and enable signal ("enable_out") for lower priority level. Interrupt enable output signal becomes low when interrupt enable input to block is low or when interrupt is requested from this level. Internal structure of each priority level decoder is shown in Figure 79.

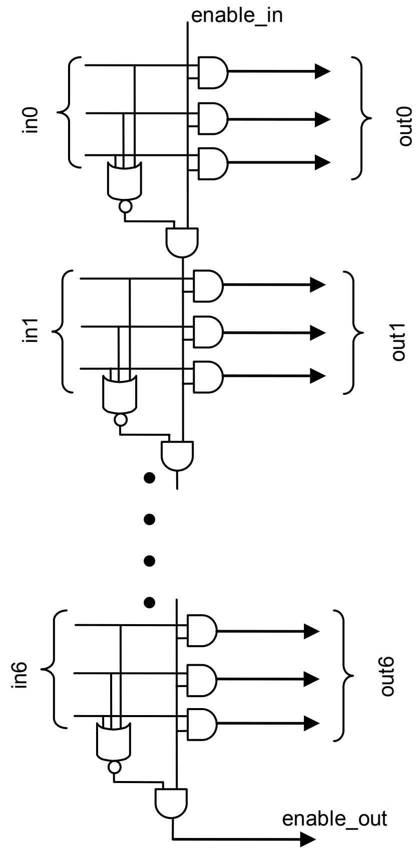


Figure 79. Priority Level Diagram

The enable input to interrupt priority block for level 3 ("int_prior3") is "ien0(7)" that enables and disables all interrupts globally and "is_reg(3)" bit. The enable input to interrupt priority block for level 2 ("int_prior2") is interrupt enable output from block for level 3 and "is_reg(2)". The enable input to interrupt priority block for level 1 ("int_prior1") is interrupt enable output

from block for level 2 and "is_reg(1)". The enable input to interrupt priority block for level 0 ("int_prior0") is interrupt enable output from block for level 1 and "is_reg(0)".

Outputs "out0" from all priority level blocks are OR-ed together and form the "int_req0". The other outputs from priority level blocks ("out1", "out2" …) are OR-ed in this same way and form signals "int_req1", "int_req2", … .

The interrupt priority structure organized as described above ensures that only one bit of "int0", "int1" … is active at this same time. Additionally when some bit is active it means that this interrupt has now the highest priority and can be requested to CPU (no interrupt with the same or higher priority is in service).

In the next step all the "int_req" ("int_req0" to "int_req6") signals are stored in flip-flops. A logical OR of all request makes the main interrupt request to the CPU and upon interrupt request stored in flip-flips the interrupt vector is generated. The ISR structure that makes this function is shown in Figure 80.

The relation between interrupt requests stored in flip-flops and interrupt vector generated to the CPU is shown in table Table 86. The exac values of interrupt vectors (INT_VECT0, INT_VECT1 etc) are defined in package or parameter file. When no interrupt request is set, the "intvect" keeps the value of the last used interrupt vector.

Table 86. Interrupt vector and interrupt request

Interrupt request source	intvect	int_req0_reg, int_req1_reg, int_req2_reg, int_req3_reg, int_req4_reg, int_req5_reg, int_req6_reg
int_vect_03	INT_VECT0	10000, 00000, 00000, 00000, 00000, 00000, 00000
int_vect_0B	INT_VECT1	00000, 10000, 00000, 00000, 00000, 00000, 00000
int_vect_13	INT_VECT2	00000, 00000, 10000, 00000, 00000, 00000, 00000
int_vect_1B	INT_VECT3	00000, 00000, 00000, 10000, 00000, 00000, 00000
int_vect_23	INT_VECT4	00000, 00000, 00000, 00000, 10000, 00000, 00000
int_vect_2B	INT_VECT5	00000, 00000, 00000, 00000, 00000, 10000, 00000
int_vect_33	INT_VECT18	00000, 00000, 00000, 00000, 00000, 00000, 10000
int_vect_43	INT_VECT6	00001, 00000, 00000, 00000, 00000, 00000, 00000
int_vect_4B	INT_VECT7	00000, 00001, 00000, 00000, 00000, 00000, 00000
int_vect_53	INT_VECT8	00000, 00000, 00001, 00000, 00000, 00000, 00000
int_vect_5B	INT_VECT9	00000, 00000, 00000, 00001, 00000, 00000, 00000
int_vect_63	INT_VECT10	00000, 00000, 00000, 00000, 00001, 00000, 00000
int_vect_6B	INT_VECT11	00000, 00000, 00000, 00000, 00000, 00001, 00000
int_vect_EB	INT_VECT29	00000, 00000, 00000, 00000, 00000, 00000, 00001
int_vect_83	INT_VECT12	00100, 00000, 00000, 00000, 00000, 00000, 00000
int_vect_8B	INT_VECT13	00000, 00100, 00000, 00000, 00000, 00000, 00000
int_vect_93	INT_VECT14	00000, 00000, 00100, 00000, 00000, 00000, 00000
int_vect_9B	INT_VECT15	00000, 00000, 00000, 00100, 00000, 00000, 00000
int_vect_A3	INT_VECT16	00000, 00000, 00000, 00000, 00100, 00000, 00000
int_vect_AB	INT_VECT17	00000, 00000, 00000, 00000, 00000, 00100, 00000
int_vect_BB	INT_VECT23	00000, 00000, 00000, 00000, 00000, 00000, 00100

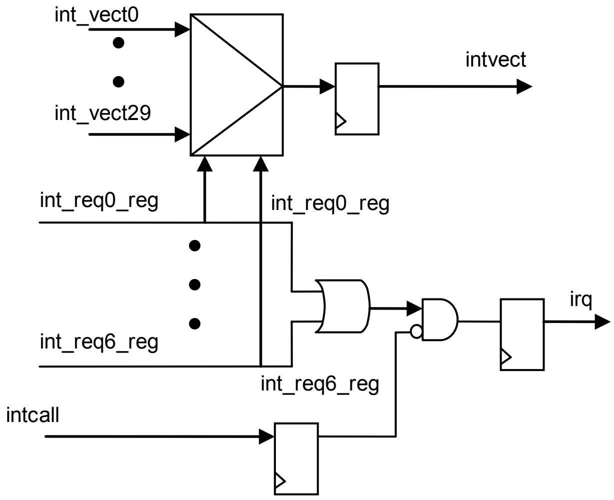


Figure 80. Interrupt Request and Interrupt Vector Generation Diagram

The ISR structure described above inserts 2 clock cycle delay between interrupt request to the ISR and interrupt request sent by ISR to CPU.

When the ISR sets interrupt request to the CPU, it responds by executing interrupt acknowledge cycle and generating the "intcall" signal. The timing of interface signals between ISR and CPU is shown in Figure 81.

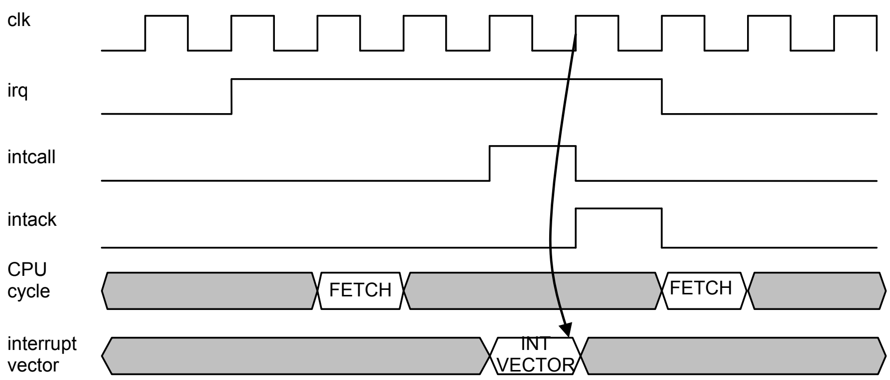


Figure 81. Interrupt Request Handshaking Timings

When the CPU sets "intcall" signal, it informs the ISR that it has accepted interrupt request and that on nearest rising edge of the clock it will latch the interrupt vector. Active "intcall" signal forces the ISR to generate the corresponding interrupt acknowledge signal (acknowledge to interrupt request source) and to update the "is_reg register". Interrupt acknowledge signals and new value of the "is_reg" are generated upon interrupt vector sent to the CPU. This solution ensures that the "is_reg" and acknowledge signal will be generated correctly according to interrupt service subroutine taken by the CPU. The hardware that implements this function is shown in Figure 82.

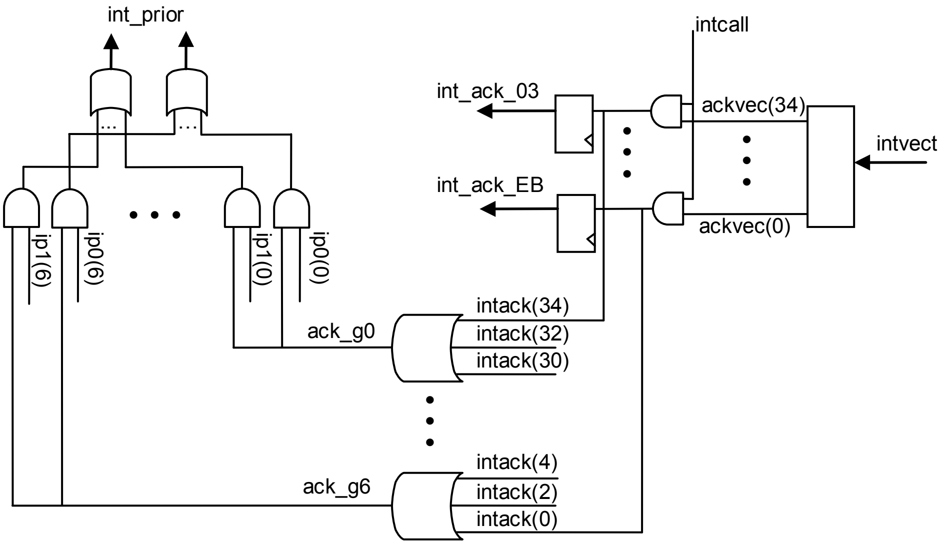


Figure 82. Interrupt acknowledge generation

The "is_reg" signal that contains information about interrupts in service (interrupts with which interrupt priority levels are in service) is updated only when "intcall" or "intreti" signals are active. When the "intcall" is active, information about new interrupt in service is added to the "is_reg". in other hand, when "intreti" signal is active, this means that some interrupt service subroutine has finished. Always the interrupt with the highest priority is finished first.

Hardware responsible for handling the "is_reg" register is shown in Figure 83.

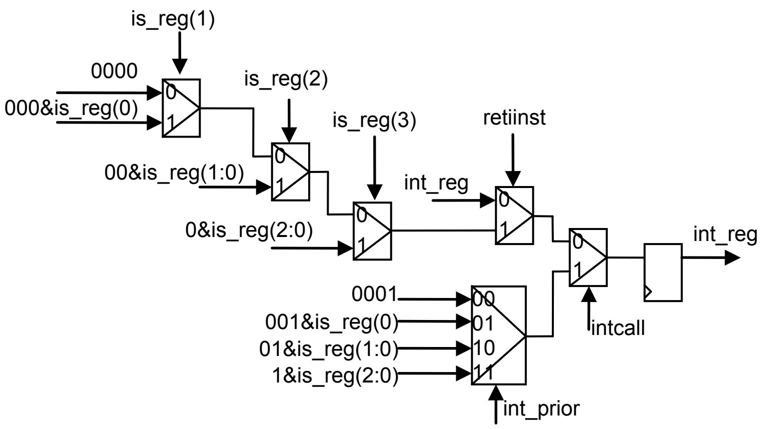


Figure 83. IS_REG Handling Logic

c) ISR Test Mode

The "isr_tm" input pin is designed to substitute the interrupt requests coming from Timer 0, 1 & 2, PCA, Serial Port 0 & 1, External Interrupt 2 shared with the SPI interrupt, and External Interrupt 7 shared with I2C interrupt. When "isr_tm" is set to 1, the corresponding interrupt requests are connected to direct inputs to the core instead of the original sources, to ease the verification process of the ISR.

d) Interrupt Connections

The table below shows the interconnections between interrupt sources (peripherals) and the inputs to the ISR module.

Table 87. Interrupt Connections

Interrupt input / vector	Source	Test mode	Description
int_vect_03	ie0	ie0	External Interrupt 0
int_vect_0B	t0_tf0	t0ff	Timer 0 overflow
int_vect_13	ie1	ie1	External Interrupt 1
int_vect_1B	tf1_gate	t1ff	Timer 1 overflow
int_vect_23	riti0_gate	rxd0ff	Serial Port 0 interrupt
int_vect_2B	tfexf2_gate	t2iff	Timer 2 overflow
int_vect_33	cfccf_gate	eciff	PCA interrupt
int_vect_43	iex7_gate	sdaiff	External Interrupt 7 / I2C interrupt
int_vect_4B	iex2_gate	misoiff	External Interrupt 2 / SPI interrupt
int_vect_53	iex3	iex3	External Interrupt 3
int_vect_5B	iex4	iex4	External Interrupt 4
int_vect_63	iex5	iex5	External Interrupt 5
int_vect_6B	iex6	iex6	External Interrupt 6
int_vect_83	riti1_gate	rxd1ff	Serial Port 1 interrupt
int_vect_8B	iex8	iex8	External Interrupt 8
int_vect_93	iex9	iex9	External Interrupt 9
int_vect_9B	iex10	iex10	External Interrupt 10
int_vect_A3	iex11	iex11	External Interrupt 11
int_vect_AB	iex12	iex12	External Interrupt 12
int_vect_BB	iex13	iex13	External Interrupt 13
int_vect_EB	rtcint	rtcint	Real Time Clock Interrupt

5.3. TIMER0

5.3.1 Overview

The **TIMER0** subcomponent contains the Timer 0 - a 16-bit register that can be configured for counter or timer operations. It can be accessed as SFRs: "th0" and "tl0" (2.4.50).

5.3.2 Pin Description

Table 88. TIMER0 Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral Clock** Clock input for all internal synchronous logic

Name	Type	Polarity Bus size	Description
rst	I	High	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
newinstr	I	High	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
Timers interface			
t0ff	I	high	Timer 0 external input
t0ack	I	high	Timer 0 interrupt acknowledge
t1ack	I	high	Timer 1 interrupt acknowledge
int0ff	I	falling/low	External interrupt 0 input
t0_tf0	O	high	Timer 0 overflow flag
t0_tf1	O	high	Timer 1 overflow flag
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"
t0_tmod	O	4	TIMER0 – related part of the "tmod" register
t0_tr0	O	High	The "tr0" flag of the "tcon" register
tl0	O	8	Timer 0 output (low-order byte)
th0	O0	8	Timer 0 output (high-order byte)

5.3.3 Block Diagram

a) Mode 0 and Mode 1

In mode 0, Timer 0 is configured as a 13-bit register ("tl0" = 5 bits, "th0" = 8 bits). The upper 3 bits of "tl0" are unchanged and should be ignored.

In mode 1 Timer 0 is configured as a 16- bit register.

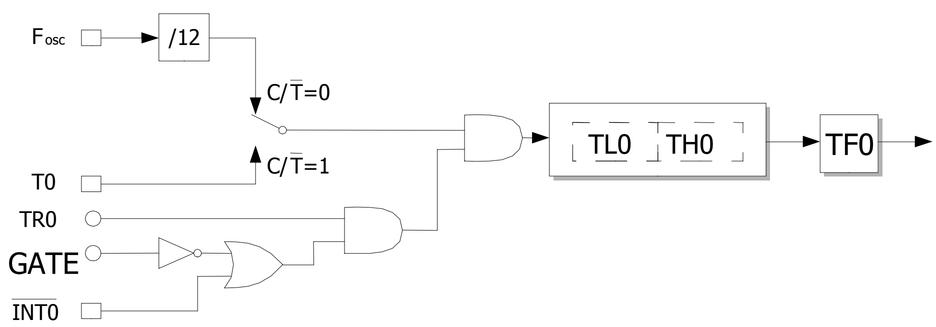


Figure 84. Timer 0 in Mode 0 and 1

b) Mode 2

In this mode the Timer 0 is configured as an 8-bit register with auto-reload.

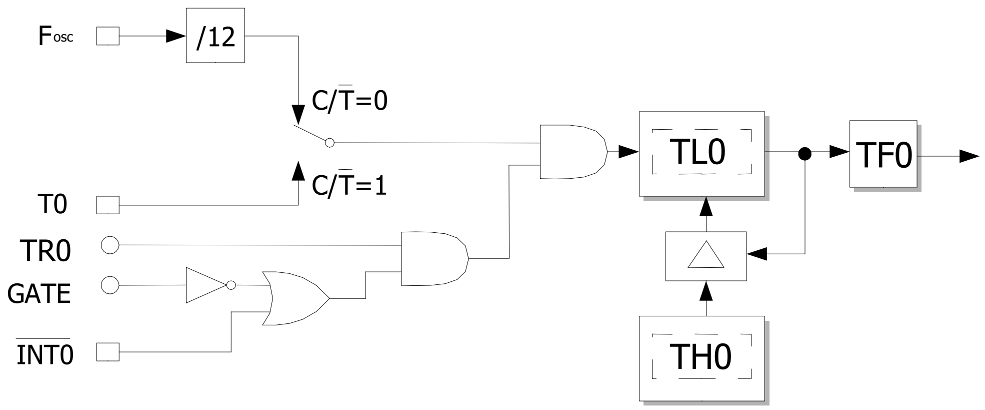


Figure 85. Timer 0 in Mode 2

c) Mode 3

In mode 3 Timer 0 is configured as one 8-bit timer/counter and one 8-bit timer. When Timer 0 works in mode 3 Timer 1 can still be used in other mode by the serial port as a baud rate generator, or application not requiring an interrupt from Timer 1.

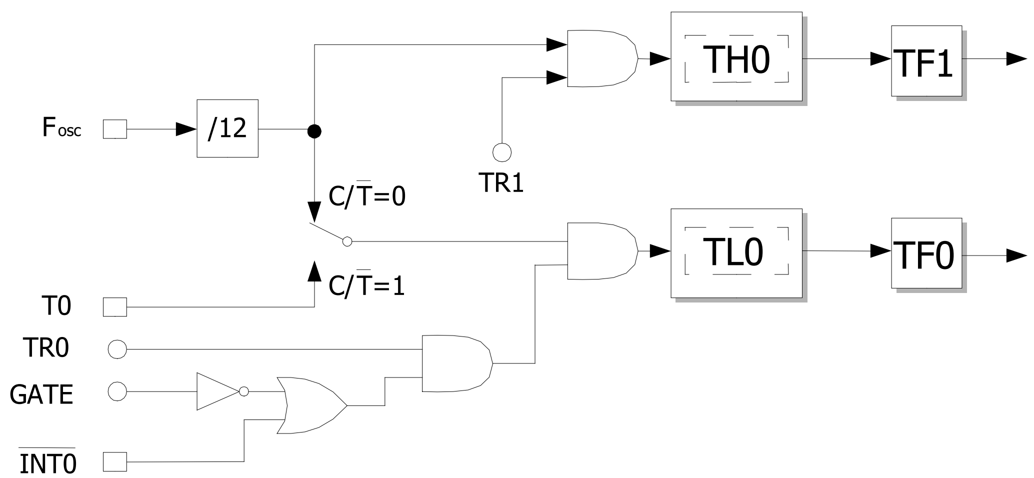


Figure 86. Timer 0 in Mode 3

5.3.4 Description

In the timer mode, the Timer 0 is incremented every 12 clock cycles, which means that it counts up after every 12 periods of the clock signal.

In the counter mode, the Timer 0 is incremented when the falling edge is detected at the corresponding input pin – "t0" for Timer 0. Since it takes 2 clock cycles to recognize a 1-to-0 event, the maximum input count rate is 1/2 of the oscillator frequency. There are no restrictions on the duty cycle, however to ensure proper recognition of 0 or 1 state, an input should be stable for at least 1 clock cycle.

Four operating modes can be selected for Timer 0. Two Special Function Registers: "tmod" (2.4.53) and "tcon" (0) are used to select the appropriate mode.

a) Timer/Counter 0 in Mode 0

This mode is invoked by setting the "tmod[1:0]"="00" flags of the "tmod" register (2.4.53).

In this mode, the count rate is derived from the "clk" input for Timer option or from the "t0" input for Counter option. The Timer option is selected by clearing "tmod[2]" flag, otherwise the Counter option is selected.

The Timer/Counter is divided in two 8 bit registers, lower and higher byte. Lower byte in this mode is additionally divided in two parts consisting of lower 5 bits and higher 3 bits (only higher 5 bits are part of the counter). This makes the Timer/Counter 0 a 13 bit counter that is incremented every 12 clock cycles, or when external signal "t0" changes its value from 1 to 0.

When Timer/Counter 0 overflows, the "tcon[5]" flag is set and interrupt is generated through "tf0" output pin. This bit is cleared when acknowledge signal ("int0ack") arrives.

The Timer/Counter may be controlled by software or hardware. The 'tcon[4]" flag must be set to run the Timer 0. Interrupt on "int0" stops counting, if appropriate gate flag "tmod[3]" is enabled.

See Figure 84.

b) Timer/Counter 0 in Mode 1

This mode is invoked by setting the "tmod[1:0]"="01" flags of the "tmod" register (2.4.53). This mode differs from Mode 0 only in that the lower byte is not divided in 5-bit and 3-bit parts, but the whole lower byte works as a counter. The Timer/Counter 0 is a 16 bit counter in mode 1.

See Figure 84.

c) Timer/Counter 0 in Mode 2

This mode is invoked by setting the "tmod[1:0]"="10" flags of the "tmod" register (2.4.53).

In this mode, the count rate is derived from the "clk" input for Timer option or from the "t0" input for Counter option. The Timer option is selected by clearing "tmod[2]" flag, otherwise the Counter option is selected.

In this mode only lower byte ("tl0") is incremented every 12 clock cycles or when external signal "t0" changes its value from 1 to 0.

In this mode the Timer/Counter works as an 8-bit reload timer/counter. When lower byte of Timer/Counter overflows, the "tcon[5]" flag is set and interrupt is generated through "tf0" output pin. This bit is cleared when acknowledge signal ("int0ack") arrives. Additionally, when the overflow occurs the new value is fetched from higher byte ("th0") to lower byte ("tl0").

The Timer/Counter may be controlled by software or hardware. The "tcon[4]" flag must be set to run the Timer 0. Interrupt on "int0" stops counting, if appropriate gate flag "tmod[3]" is enabled.

See Figure 85.

d) Timer/Counter 0 in Mode 3

This mode is invoked by setting the "tmod[1:0]"=11 flag of "tmod" register (2.4.53).

In this mode, the count rate for lower byte is derived from the "clk" input for Timer option or from the "t0" input for Counter option, but the count rate for higher byte is only derived from the "clk". The Timer option is selected by clearing "tmod[2]" flag, otherwise the Counter option is selected.

In this mode the lower byte ("tl0") is incremented every 12 clock cycles or when external signal "t0" changes its value from 1 to 0. The higher byte ("th0") is incremented every 12 clock cycles.

When the lower byte of Timer/Counter overflows, the "tcon[5]" flag is set and interrupt is generated through "tf0" output pin. When the higher byte overflows, the "tcon[7]" flag is set

and interrupt is generated through “tf1” output pin. These bits are cleared when appropriate acknowledge signals (“int0ack”, “int1ack”) arrive, respectively.

In this mode the lower byte of Timer/Counter 0 is controlled by ‘tcon[4]” flag which must be set to enable Timer operation, and by the “int0” input which stops counting when forced to 0 while the “tmod[3]” flag is set.

The higher byte is controlled only by “tcon[6]” flag which enables counting when set.

5.4. TIMER1

5.4.1 Overview

The **TIMER1** subcomponent contains Timer 1 – a 16-bit register that can be configured for counter or timer operations. It can be accessed as SFRs:“th1”, “tl1” (2.4.51).

5.4.2 Pin Description

Table 89. TIMER1 Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	High	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
newinstr	I	High	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
Timers interface			
t1ff	I	high	Timer 1 external input
t1ack	I	high	Timer 1 interrupt acknowledge
int1ff	I	falling/low	External interrupt 1 input
t1_tf1	O	high	Timer 1 overflow flag
t1ov	O	high	Timer 1 overflow output
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by “sfraddr”
t1_tmod	O	4	TIMER1 – related part of the “tmod” register
t1_tr1	O	High	The “tr1” flag of the “tcon” register
tl1	O	8	Timer 1 output (low-order byte)
th1	O	8	Timer 1 output (high-order byte)

5.4.3 Block Diagram

a) Mode 0 and Mode 1

In mode 0, Timer 1 is configured as a 13-bit register (”tl1” = 5 bits, ”th1” = 8 bits). The upper 3 bits of ”tl1” are unchanged and should be ignored.

In mode 1 Timer 1 is configured as a 16- bit register.

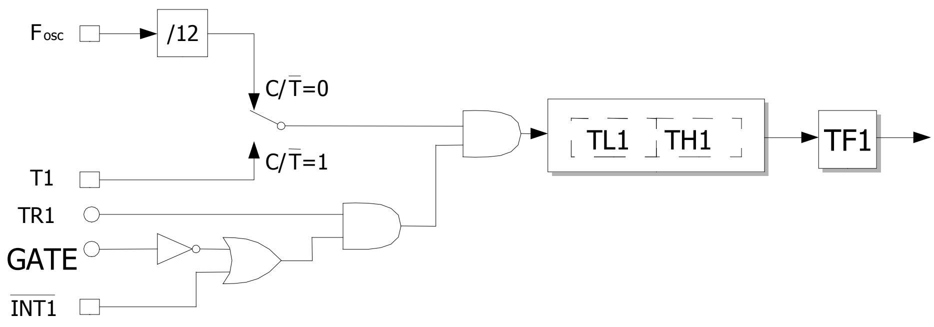


Figure 87. Timer 1 in Mode 0 and 1

b) Mode 2

In this mode the TIMER1 is configured as an 8-bit register with auto-reload.

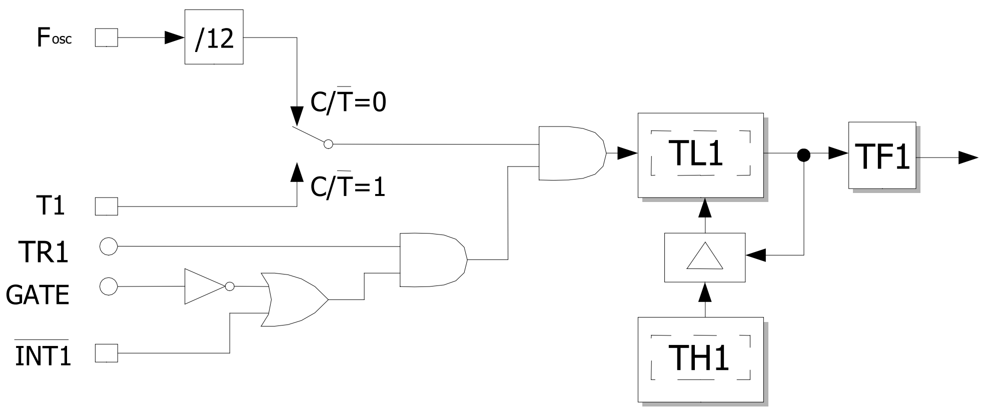


Figure 88. Timer 1 in Mode 2

c) Mode 3

In mode 3 Timer 1 is stopped.

5.4.4 Description

In the timer mode, the Timer 1 is incremented every 12 clock cycles, which means that it counts up after every 12 periods of the clock signal.

In the counter mode, the Timer 1 is incremented when the falling edge is detected at the corresponding input pin – “t1” for Timer 1. Since it takes 2 clock cycles to recognize a 1-to-0 event, the maximum input count rate is 1/2 of the oscillator frequency. There are no restrictions on the duty cycle, however to ensure proper recognition of 0 or 1 state, an input should be stable for at least 1 clock cycle.

Four operating modes can be selected for Timer 1. Two Special Function Registers: “tmod” (2.4.53) and “tcon” (0) are used to select the appropriate mode.

a) Timer/Counter 1 in Mode 0

This mode is invoked by setting the "tmod[5:4]"="00" flags of the "tmod" register (2.4.53).

In this mode, the count rate is derived from the "clk" input for Timer option or from the "t1" input for Counter option. The Timer option is selected by clearing "tmod[6]" flag, otherwise the Counter option is selected.

The Timer/Counter 1 is divided in two 8 bit registers, lower and higher byte. Lower byte in this mode is additionally divided in two parts consisting of lower 5 bits and higher 3 bits (only higher 5 bits are part of the counter). This makes the Timer/Counter 1 a 13 bit counter that is incremented every 12 clock cycles or when external signal "t1" changes its value from 1 to 0.

When Timer/Counter 1 overflows, the "tcon[7]" flag is set and interrupt is generated through "tf1" output pin. This bit is cleared when acknowledge signal ("int1ack") arrives.

The Timer/Counter 1 may be controlled by software or hardware. The 'tcon[6]" flag must be set to run the Timer 1. Interrupt on "int1" stops counting, if appropriate gate flag "tmod[7]" is enabled.

See Figure 87.

b) Timer/Counter 1 in Mode 1

This mode is invoked by setting the "tmod[5:4]"="01" flags of the "tmod" register (2.4.53). This mode differs from Mode 0 only in that the lower byte is not divided in 5-bit and 3-bit parts, but the whole lower byte works as a counter. The Timer/Counter 1 is a 16 bit counter in mode 1.

See Figure 87.

c) Timer/Counter 1 in Mode 2

This mode is invoked by setting the "tmod[5:4]"="10" flags of the "tmod" register (2.4.53).

In this mode, the count rate is derived from the "clk" input for Timer option or from the "t1" input for Counter option. The Timer option is selected by clearing the "tmod[6]" flag, otherwise the Counter option is selected.

In this mode the Timer/Counter works as an 8-bit reload timer/counter. Only lower byte ("tl1") is incremented every 12 clock cycles or when external signal "t1" changes its value from 1 to 0.

When lower byte of Timer/Counter overflows, the "tcon[7]" flag is set and interrupt is generated through "tf1" output pin. This bit is cleared when acknowledge signal ("int1ack") arrives. Additionally when the overflow occurs the new value is fetched from higher byte ("th1") to lower byte ("tl1").

The Timer/Counter may be controlled by software or hardware. The "tcon[6]" flag must be set to run the Timer 1. Interrupt on "int1" stops counting, if appropriate gate flag "tmod[7]" is enabled.

SeeFigure 88.

d) Timer/Counter 1 in Mode 3

This mode is invoked by setting the "tmod[5:4]"=11 flag of "tmod" register (2.4.53).
In this mode the Timer/Counter 1 is disabled (only Timer/Counter 0 can operate in mode 3).

5.5. TIMER2_251

5.5.1 Overview

The **TIMER2_251** subcomponent contains a 16-bit register that can be configured for counter or timer operations. It can be accessed as SFRs:"th2", "tl2" (2.4.52) and controlled by "t2con"(2.4.47) and "t2mod"(2.4.48) registers. There are also "rcap2h" and "rcap2l" (2.4.31) registers for capture an reload functions.

5.5.2 Pin Description

Table 90. TIMER2_251 Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	high	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
newinstr	I	high	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
percycle	I	4	**Peripheral cycles** Current number of peripheral cycle
Timer 2 inputs			
t2iff	I	high/fall	Timer 2 external gate/edge input
t2exff	I	fall	Timer 2 capture trigger
Timer 2 outputs			
t2o	O	-	Timer 2 clock
t2ov	O	high	Timer 2 overflow (for Serial 0)
tf2	O	high	Timer 2 overflow flag
exf2	O	high	Timer 2 external signal
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	high	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"
rclk	O	high	Recieve clock bit select
tclk	O	high	Transmit clock bit select
exen2	O	high	External interrupt 2 enable flag
tr2	O	-	Run control bit
ct2	O	-	Counter/Timer Select
cprl2	O	-	Capture/Reload Select
t2oe	O	-	Output enable bit

Name	Type	Polarity Bus size	Description
dcen	O	-	Down count enable bit
tl2	O	8	Timer 2 output (low-order byte)
th2	O	8	Timer 2 output (high-order byte)
rcap2l	O	8	Reload/Capture Register 0 (low-order byte)
rcap2h	O	8	Reload/Capture Register 0 (high-order byte)

5.6. TIMER2_515

5.6.1 Overview

The **TIMER2** subcomponent is composed of TIMER2 that can be configured for either counter or timer operations, and the Compare/Capture Unit which is a sub-component of TIMER2.

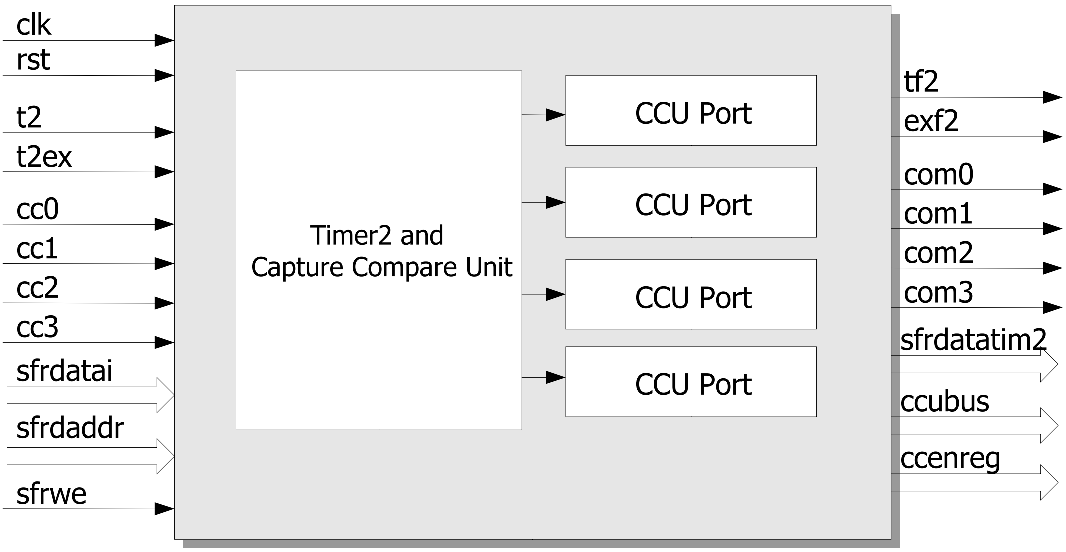


Figure 89. TIMER2 Symbol

5.6.2 Pin Description

Table 91. TIMER2 Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	high	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
newinstr	I	high	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
Timer 2 inputs			
t2ff	I	high/fall	Timer 2 external gate/edge input

Name	Type	Polarity Bus size	Description
t2exff	I	fall	Timer 2 capture trigger
Comare/Capture interface			
ccff(0)	I	rise/fall	compare/capture 0 input
ccff(1)	I	rise	compare/capture 1 input
ccff(2)	I	rise	compare/capture 2 input
ccff(3)	I	rise	compare/capture 3 input
com	O	4	compare 0..3 output
T2CON interface			
tf2	O	high	Timer 2 overflow signal
exf2	O	high	Timer 2 external signal
ccubus	O	4	Compare output bus (direct output of the core)
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	high	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"
tl2	O	8	Timer 2 output (low-order byte)
th2	O	8	Timer 2 output (high-order byte)
exen2	O	high	External interrupt 2 enable flag
crcl	O	8	Compare/Capture Register 0 (low-order byte)
crch	O	8	Compare/Capture Register 0 (high-order byte)
t2con	O	8	Timer 2 Control Register
ccen	O	8	Compare/Capture Enable Register
ccl1	O	8	Compare/Capture Register 1 (low-order byte)
cch1	O	8	Compare/Capture Register 1 (high-order byte)
ccl2	O	8	Compare/Capture Register 2 (low-order byte)
cch2	O	8	Compare/Capture Register 2 (high-order byte)
ccl3	O	8	Compare/Capture Register 3 (low-order byte)
cch3	O	8	Compare/Capture Register 3 (high-order byte)

Table 92. CCU_PORT Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	high	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
compare	I	high	Compare signal In compare mode 0 directly controlled port

Name	Type	Polarity Bus size	Description
ov	I	high	Timer 2 overflow In compare mode 0 directly controlled port
cocahl	I	2	Compare function enable - ccen
t2cm	I	-	Compare mode bit -t2con(2): when 0 -mode 0 when 1 -mode 1
pout	O	-	Compare output to Port 1
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"

5.6.3 Block Diagram

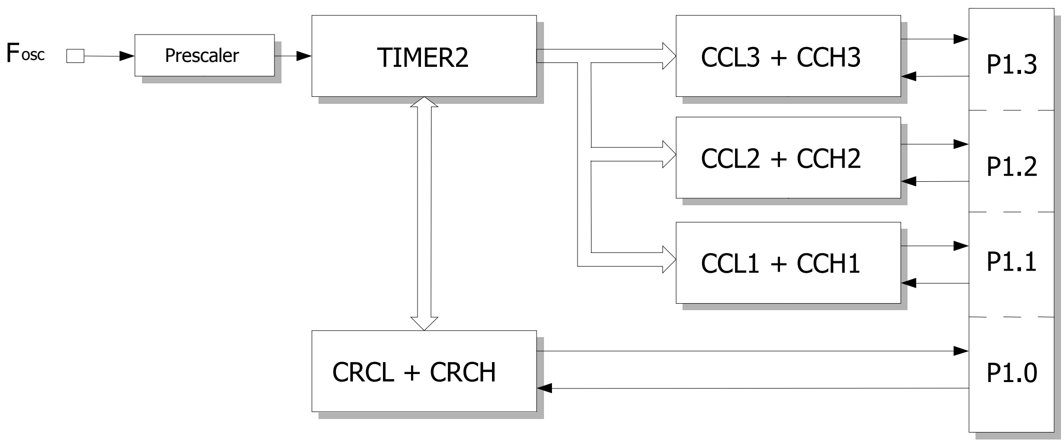


Figure 90. TIMER2 Block Diagram

5.6.4 Timer 2 Description

The Timer 2 can operate as timer, event counter, or gated timer.

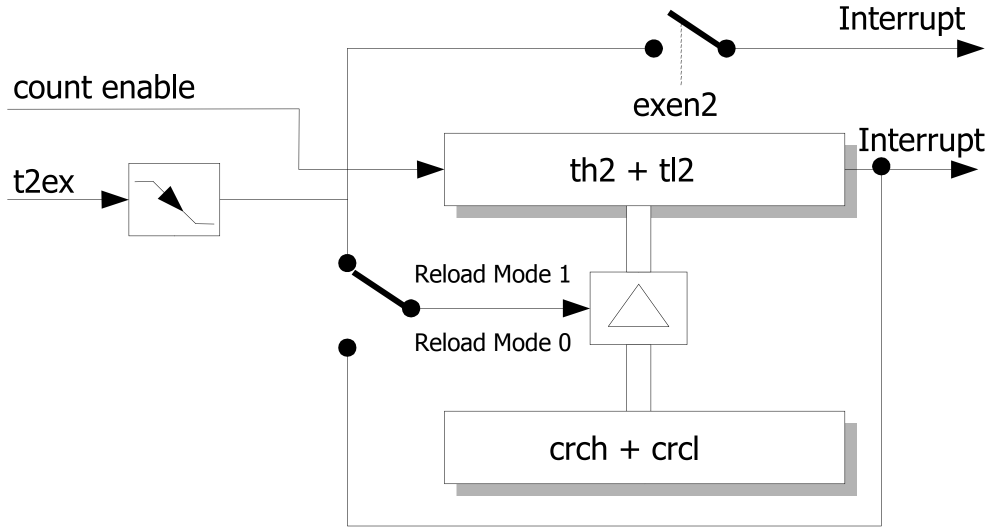


Figure 91. Timer 2 in Reload Mode

a) Timer Mode

This mode is invoked by setting the "t2i0"=1 and "t2i1"=0 flags of "t2con" register (2.4.47).

In this mode, the count rate is derived from the "clk" input.

The Timer 2 is incremented every 12 or 24 clock cycles depending on the 2:1 prescaler. The prescaler mode is selected by bit "t2ps" of "t2con" register (2.4.47). When "t2ps"=0, the timer counts up every 12 clock cycles, otherwise every 24 cycles.

b) Event Counter Mode

This mode is invoked by setting the "t2i0"=0 and "t2i1"=1 flags of "t2con" register (2.4.47).

In this mode the Timer 2 is incremented when external signal "t2" changes its value from 1 to 0. The "t2" input is sampled at every rising edge of the clock. The Timer 2 is incremented in the cycle following the one in which the transition was detected. The maximum count rate is ½ of the clock frequency.

c) Gated Timer Mode

This mode is invoked by setting the "t2i0"=1 and "t2i1"=1 flags of "t2con" register (2.4.47).

In this mode the Timer 2 is incremented every 12 or 24 clock cycles (depending on "t2ps" flag) but additionally it is gated by external signal "t2". When "t2"=0, the Timer 2 is stopped. The "t2" input is sampled into a flip-flop and then it blocks the Timer 2 incrementation.

d) Timer 2 Reload

A 16-bit reload from the "crc" register (2.4.8) can be executed in two modes:

- Reload Mode 0: Reload signal is generated by Timer 2 overflow (auto reload)
- Reload Mode 1: Reload signal is generated by negative transition at the corresponding input pin "t2ex".

5.6.5 Compare Function

The Compare/Capture Unit consists of four registers: "cc1", "cc2", "cc3" (2.4.5) and "crc" (2.4.8). Each of these registers can be configured to work in comparator mode. in this mode the value stored in register is compared with the contents of the TIMER2. The comparators outputs drive four low ordered bits of port "p1" ("p1.0" … "p1.3"), where:

- "p1.0" is output of the comparator associated with the register "crc" ("ccubus.0")
- "p1.1" is the output of comparator associated with the register "cc1" ("ccubus.1")
- "p1.2" is the output of comparator associated with the register "cc2" ("ccubus.2")
- "p1.3" is the output of comparator associated with the register "cc3" ("ccubus.3")

There are two compare modes selected by bit "t2cm" in "t2con" register.

a) Compare Mode 0

The Compare Mode 0 is invoked by setting bit "t2cm"=0 of "t2con" register (2.4.47).

In mode 0, when the value in Timer 2 equals the value of the compare register, the comparator output changes from low to high. It goes back low on timer 2 overflow.

In this mode writing to port 1 ("p1") will have no effect, because the input line from the internal bus and the write-to-register line are disconnected. Figure below illustrates the function of compare mode 0.

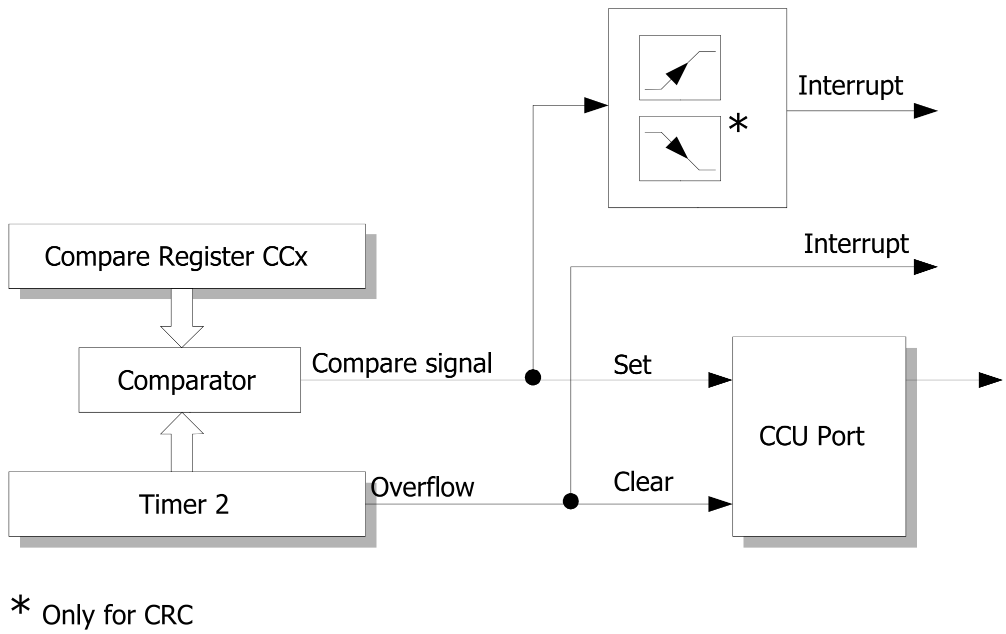


Figure 92. Timer 2 in Compare Mode 0

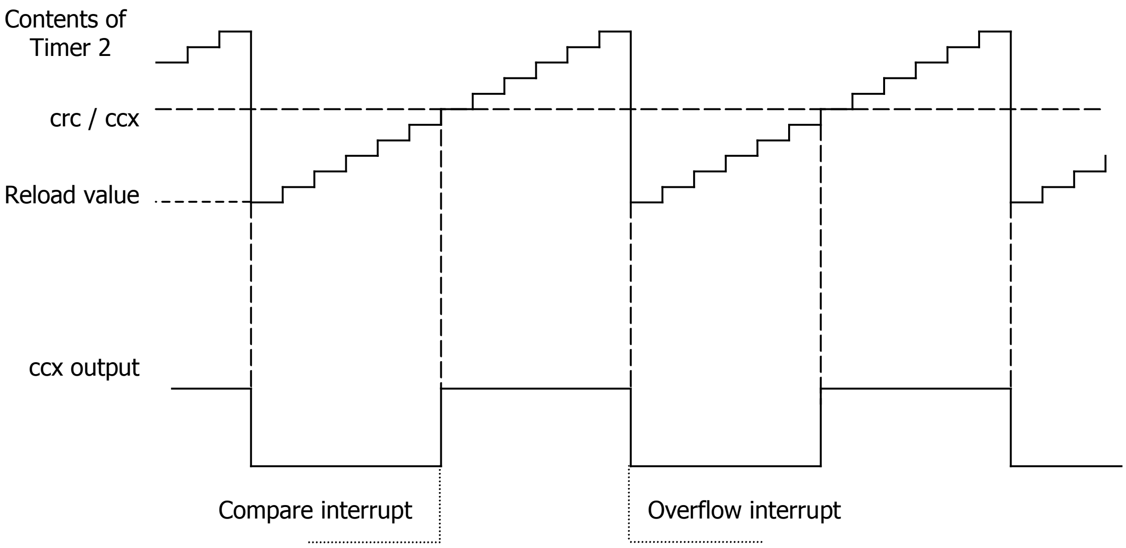
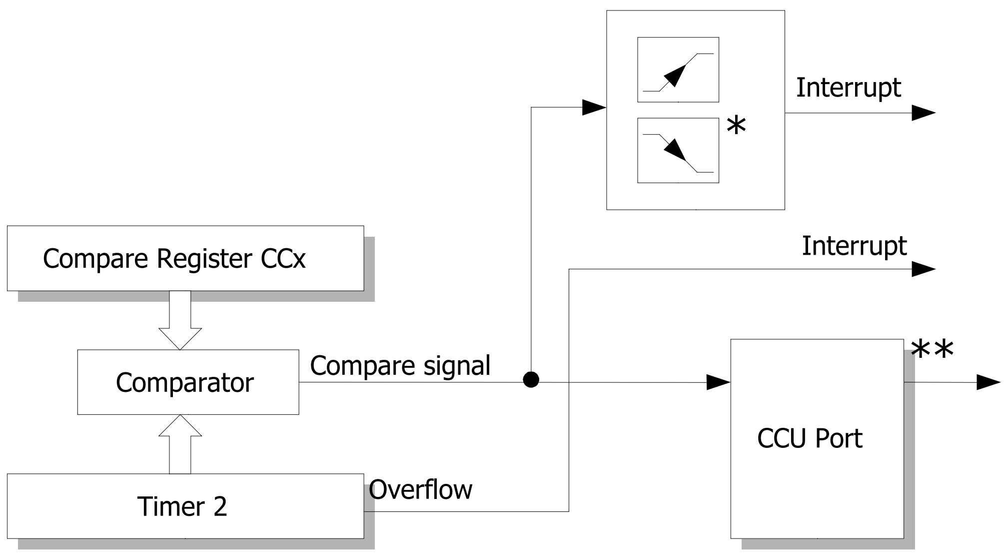


Figure 93. Compare Mode 0 Operation

b) Compare Mode 1

The Compare Mode 1 is invoked by setting bit "t2cm"=1 of "t2con" register (2.4.47).

In compare mode 1, the transition of the output signal can be determined by software. A Timer 2 overflow causes no output change. in this mode both transitions of output signal can be controlled. Figure below shows a functional diagram of a register/port configuration in compare mode 1. in compare mode 1 the value is written first to the "Shadow Register", and when the compare signal goes active this value is transferred to the output register.



* Only for CRC

** Shadow register is used

Figure 94. Timer 2 in Compare Mode 1

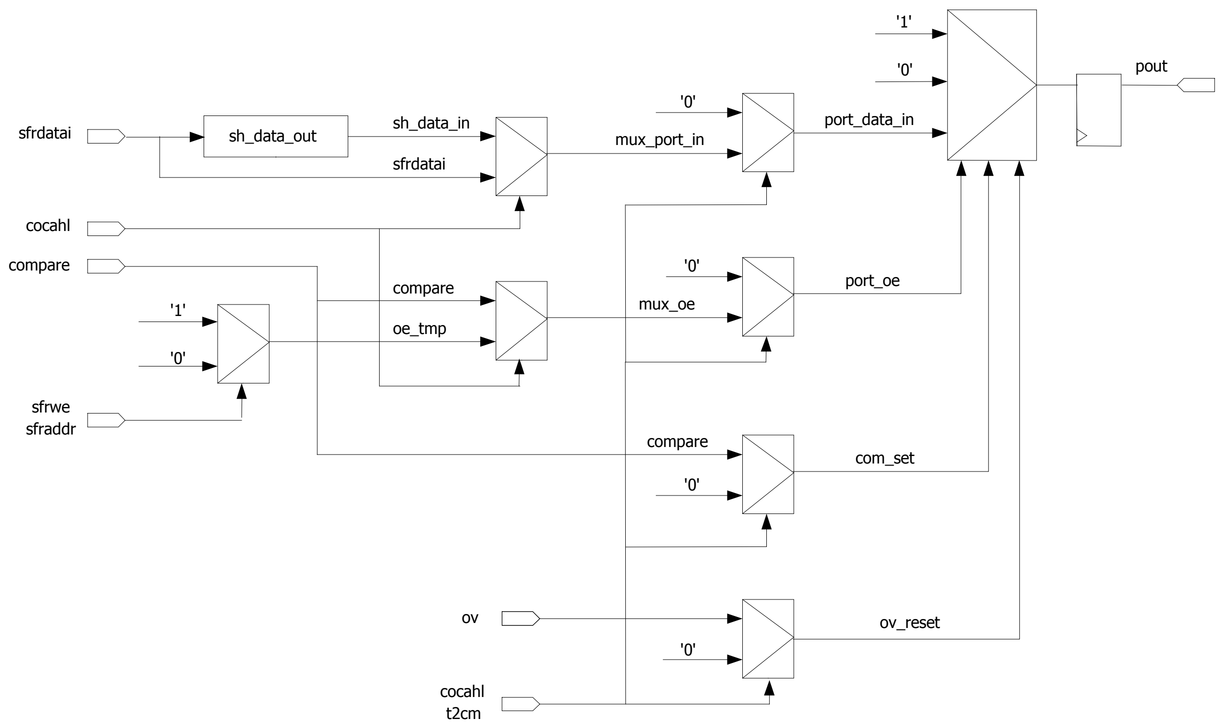


Figure 95. CCU Port diagram

5.6.6 Capture Function Description

Each of four 16-bit CCU registers can be configured to work in capture mode.

In this mode the actual timer/counter contents are saved into CCU register upon an external event (mode 0) or a software write operation (mode 1).

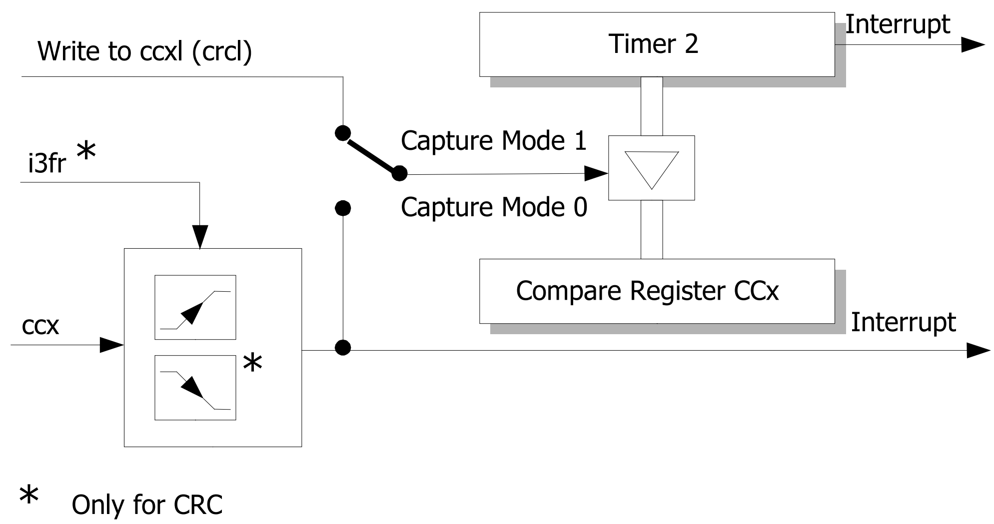


Figure 96. Timer 2 in Capture Mode

a) Capture Mode 0

In mode 0 capturing of Timer 2 contents is executed when:

- rising edge is detected on input "cc1" (for "cc1" register working in capture mode)
- rising edge is detected on input "cc2" (for "cc2" register working in capture mode)

- rising edge is detected on input "cc3" (for "cc3" register working in capture mode)
- rising or falling edge is detected on input "cc0", depending on bit "i3fr" (for "crc" register working in capture mode)

The timer 2 contents will be latched into appropriate capture register. in this mode no interrupt request will be generated.

b) Capture Mode 1

In mode 1 capture of Timer 2 is caused by any write into the low-ordered byte of the dedicated capture register. The value written to capture register is irrelevant for this function. The Timer 2 contents will be latched into appropriate capture register. in this mode no interrupt request will be generated.

5.7. PCA

5.7.1 Overview

The PCA (Programmable Counter Array) subcomponent is composed of a 16-bit Timer/Counter and five Compare/Capture modules. It performs a lot of timing and counting operations including Pulse Width Modulations and Watchdog Timer.

5.7.2 Pin Description

Table 93. PCA Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	high	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
newinstr	I	high	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
percycle	I	4	**Peripheral cycles** Current number of peripheral cycle
t0ov	I	high	**Timer 0 overflow flag**
idle	I	high	**Idle Mode Flag**
PCA input			
eciff	I	high/fall	**PCA external input**
Comare/Capture interface			
cexiff	I	5(rise/fall)	**Compare/Capture inputs**
cexo	O	5	**Compare outputs**
PCA Watchdog output			
pcawdts	O	high	**PCA Watchdog reset signal**
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs

Name	Type	Polarity Bus size	Description
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	high	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"
cidl	O	-	**Idle Mode Control**
wdte	O	-	**PCA Watchdog Enable**
cps	O	2	**Input of Timer/Counter Select**
ecf	O	-	**Interrupt from Timer/Counter Enable**
cf	O	-	**Timer/Counter Overflow Flag**
cr	O	-	**Timer/Counter Run Control**
ccf	O	5	**Compare/Capture Flags**
ecom	O	5	**Compare Modes of Compare/Capture Modules**
capp	O	5	**Positive Capture Mode of Compare/Capture Modules**
capn	O	5	**Negative Capture Mode of Compare/Capture Modules**
mat	O	5	**Match of Compare/Capture Modules**
tog	O	5	**Toggle of Compare/Capture Modules**
pmw	O	5	**Pulse Width Modulation Mode of Compare/Capture Modules**
eccf	O	5	**Interrupt from Compare/Capture Modules Enable**
cl	O	8	**Timer/Counter Register (low-order byte)**
ch	O	8	**Timer/Counter Register (high-order byte)**
ccap0l	O	8	**Compare/Capture Register 0 (low-order byte)**
ccap0h	O	8	**Compare/Capture Register 0 (high-order byte)**
ccap1l	O	8	**Compare/Capture Register 1 (low-order byte)**
ccap1h	O	8	**Compare/Capture Register 1 (high-order byte)**
ccao2l	O	8	**Compare/Capture Register 2 (low-order byte)**
ccap2h	O	8	**Compare/Capture Register 2 (high-order byte)**
ccap3l	O	8	**Compare/Capture Register 3 (low-order byte)**
ccap3h	O	8	**Compare/Capture Register 3 (high-order byte)**
ccap4l	O	8	**Compare/Capture Register 4 (low-order byte)**
ccap4h	O	8	**Compare/Capture Register 4 (high-order byte)**

5.7.3 Description

The main part of PCA component is a 16-bit Timer/Counter which serves as time base and event counter for Compare/Capture modules. The value of Timer/Counter is accessible by the "ch/cl" SFR register pair and the values of five Compare/Capture modules can be read or write by SFR register pairs: "ccap0h/ccap0l", "ccap1h/ccap1l", "ccap2h/ccap2l", "ccap3h/ccap3l" and "ccap4h/ccap4l". Control function, mode select and status reading are realized by the "cmod" and the "ccon" registers for Timer/Counter and PCA interrupts, and by the "ccapm0", "ccapm1", "ccapm2", "ccapm3" and "ccapm4" for Compare/Capture modules.

The Timer/Counter and Compare/Capture modules interrupt sources share a single interrupt vector. The PCA interrupt request to ISR is realized as a logic OR of all PCA interrupt sources. Detailed PCA interrupt cause detection is possible by the "ccon" register reading, where

compare and capture events in Compare/Capture modes and Timer/Counter overflows set proper bits. The PCA interrupt request is generated when:

- the "cf" overflow flag in the "ccon" register and the "ecf" interrupt enable bit in the "cmod" register are set,
- the "ccf0" compare/capture flag in the "ccon" register and the "eccf0" enable bit in the "ccapm0"register are set,
- the "ccf1" compare/capture flag in the "ccon" register and the "eccf1" enable bit in the "ccapm1"register are set,
- the "ccf2" compare/capture flag in the "ccon" register and the "eccf2" enable bit in the "ccapm2"register are set,
- the "ccf3" compare/capture flag in the "ccon" register and the "eccf3" enable bit in the "ccapm3"register are set,
- the "ccf4" compare/capture flag in the "ccon" register and the "eccf4" enable bit in the "ccapm4"register are set.

All interrupt flags ("cf", "ccf0", "ccf1", "ccf2", "ccf3", "ccf4") in the "ccon" register can be cleared by software only.

a) Timer/Counter

A 16-bit Timer/Counter is composed of two 8-bit Special Function Registers: "ch" (high byte) and "cl" (low byte). These registers can be read anytime, but writing is enabled when the "cr" bit in the "ccon" register is cleared (equal '0') only (Timer/Counter is stopped). The "cps1" and "cps0" bits in the "cmod" register determine which one of the four following signals increment Timer/Counter lower byte ("cl"):

- clock pulse with Fclkper/12 frequency when "cps1,cps0" = 00,
- clock pulse with Fclkper/4 frequency when "cps1,cps0" = 01,
- Timer 0 overflow (sampled with Fclkper/12 frequency) when "cps1,cps0" = 10,
- the "eciff" falling edge on external input (sampled with Fclkper/4 frequency – the maximum input frequency is Fclkper/8) when "cps1,cps0" = 11,

The "cl" register overflow increments the "ch" register after 2 cycles of "clkper" clock. The "ch" register overflow sets the "cf" flag in the "ccon" register. The "cf" flag generate interrupt request to ISR when the "ecf" bit in the "cmod" register is set. The "cr" bit in the "ccon" register starts and stop Timer/Counter. When "cidl" flag in the "cmod" register is set (equal '1'), the Timer/Counter works in IDLE mode (when the "pcon.0" bit is set).

b) Capture Mode of Compare/Capture modules

Each of five Compare/Capture modules register pair ("ccap0h/ccap0l", "ccap1h/ccap1l" , "ccap2h/ccap2l" , "ccap3h/ccap3l" and "ccap4h/ccap4l") can be configured to work in capture mode.

In this mode the actual Timer/Counter contents are saved into the registers upon an external event. This capturing is executed when:

- rising edge is detected on inputs: "cexi0", "cexi1", "cexi2", "cexi3" or "cexi4" (when appropriately: the "capp0" bit in the "ccapm0" register is set, the "capp1" bit in the "ccapm1" register is set, the "capp2" bit in the "ccapm2" register is set, the "capp3" bit in the "ccapm3" register is set or the "capp4" bit in the "ccapm4" register is set),
- falling edge is detected on inputs: "cexi0", "cexi1", "cexi2", "cexi3" or "cexi4" (when appropriately: the "capn0" bit in the "ccapm0" register is set, the "capn1" bit in the "ccapm1" register is set, the "capn2" bit in the "ccapm2" register is set, the "capn3" bit in the "ccapm3" register is set or the "capn4" bit in the "ccapm4" register is set)
- rising or falling edge is detected on inputs: "cexi0", "cexi1", "cexi2", "cexi3" or "cexi4" (when appropriately: the "capp0" and the "capn0" bits in the "ccapm0" register are set, the "capp1" and the "capn1" bits in the "ccapm1" register are set, the "capp2" and the "capn2" bits in the "ccapm2" register are set, the "capp3" and the "capn3" bits in the "ccapm3" register are set or the "capp4" and the "capn4" bits in the "ccapm4" register are set)

When Timer/Counter contents is latched into appropriate capture registers, then the proper "ccf*n*" bit in the "ccon" register is set. When the corresponding interrupt enable bit ("eccf*n*") in the "ccapm*n*" register is set, interrupt request will be generated.

c) Software Timer and High-speed Output Modes of Compare/Capture modules

The Compare/Capture modules can be configured to work in comparator modes. in this modes the value stored in register is compared with the contents of the Timer/Counter. The comparators outputs set corresponding "ccf*n*" flag (Software Timer Mode) and optionally drive five PCA Compare/Capture outputs ("cexo0" … "cexo4") (High-speed Output Mode), where:

- "cexo0" is output of the comparator associated with the register pair "ccap0h/ccap0l",
- "cexo1" is the output of comparator associated with the register pair "ccap1h/ccap1l",
- "cexo2" is the output of comparator associated with the register pair "ccap2h/ccap2l",
- "cexo3" is the output of comparator associated with the register pair "ccap3h/ccap3l",
- "cexo3" is the output of comparator associated with the register pair "ccap4h/ccap4l",

When the "ecom*n*" and the "mat*n*" bits in the "ccapm*n*" register are set, a match between the Timer/Counter registers and Compare/Capture registers set appropriate Compare/Capture flag ("ccf*n*" bit in the "ccon" register), next if the corresponding "eccf*n*" bit in the "ccapm*n*" register is set, the interrupt request is generated.

The High-speed Output Mode is selected by setting the "tog*n*" bit ("ecom*n*" = 1, "mat*n*" = 1) in the "ccapm*n*" register. In this mode an output signal (on the "cexo*n*" pins) is generated by toggling when a match occurs. By setting or clearing the "cexo*n*" bit by software ("cexo0".."cexo4" outputs correspond with "p1.3".."p1.7" bits in the "p1" register), the user determine initial state of the "cexo*n*" bit.

d) Watchdog Timer Mode of Compare/Capture modules

The PCA provides a programmable Watchdog Timer (WDT) as a mode option in Compare/Capture module 4. This mode is selected by setting the "ecom4" and the "mat4" bits in the "ccapm4" register and the "wdte" bit in the "cmod' register. The PCA WDT generates device reset when the value in the "ch/cl" registers and the value in "ccap4h/ccap4l" are equal.

e) Pulse Width Modulation (PWM) Mode of Compare/Capture modules

The Compare/Capture modules can be programmed as up to five independent pulse width modulators. This mode is select by setting the "ecom*n*" and the "pwm*n*" bits in the "ccapm*n*" register.
If PWM mode is selected, the "cl" register is compared with the "ccap*n*l" register. When "cl"<"ccap*n*l", the state of the "cexo*n*" output is low. When "cl"=>"ccap*n*l", the state of the "cexo*n*" output is high (until "cl" overflows from FFh to 00h).

5.8. SERIAL0

5.8.1 Overview

The SERIAL0 provides a flexible full-duplex synchronous/asynchronous receiver/transmitter. It can operate in four modes (one synchronous and three asynchronous). The Serial_0 is buffered at the receive side, i.e. it can reveive new data while the previously received is not damaged in the receive register until the completion of the 2nd transfer. The Serial_0 is fully compatible with the standard 80251 serial channel.

5.8.2 Pin Description

Table 94. SERIAL0 Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	High	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
Control signals			
newinstr	I	High	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
t1ov	I	High	Timer 1 overflow
Serial Port Interface			
rxd0ff	I	-	Serial Port 0 receive data input
txd0	O	-	Serial Port 0 transmit data output
rxd0o	O	-	Serial Port 0 mode 0 receive clock output
Interrupts			
ri0	O	High	Serial Port 0 receive flag
ti0	O	High	Serial Port 0 transmit flag
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"
s0mod	O	2	"Baud Rate Doubler" and "S0CON.7 select" bits of the PCON register
s0con	O	8	Serial 0 Control Register
s0buf	O	8	Serial 0 Data Buffer
s0addr	O	8	Serial 0 Slave Address Register
s0aden	O	8	Serial 0 Slave Address Mask Register

5.8.3 Block Diagram

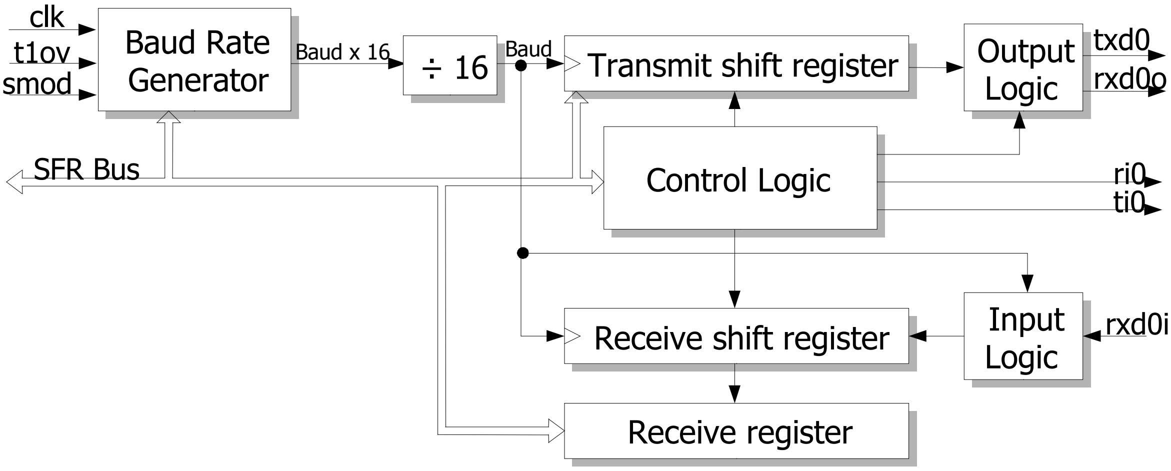


Figure 97. SERIAL0 Block Diagram

5.8.4 Description

a) The Serial Port 0 Baud Rate Generation

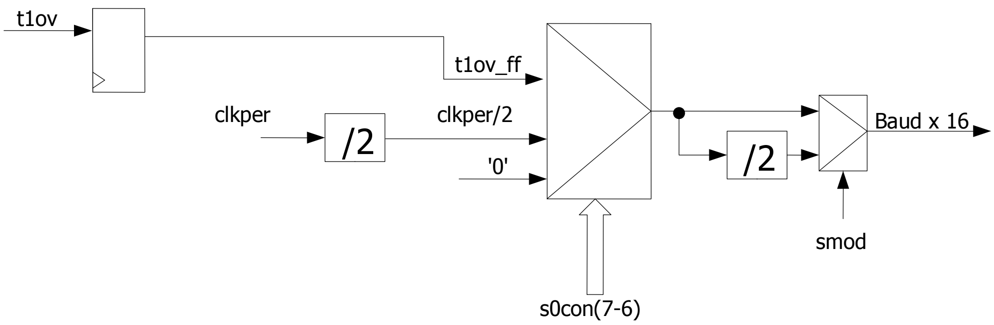


Figure 98. SERIAL0 Baud rate generation diagram

b) The Serial Port 0 Operating Modes

- **Mode 0**

In mode 0 the Serial Port 0 operates as synchronous transmitter/receiver. The "txd0" outputs the shift clock. The "rxd0o" outputs data and the "rxd0i" inputs data. 8 bits are transmitted with LSB first. The baud rate is fixed at 1/12 of the main clock frequency. Reception is started by setting the"ren0" = 1 flag "s0con" register (2.4.34), and clearing the "ri0" flag. Transmission is started by writing data to "s0buf" register.

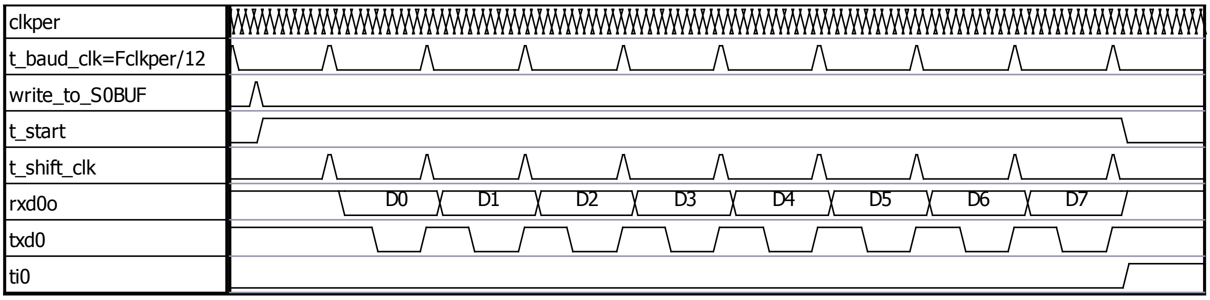


Figure 99. SERIAL0 Transmission in Mode 0

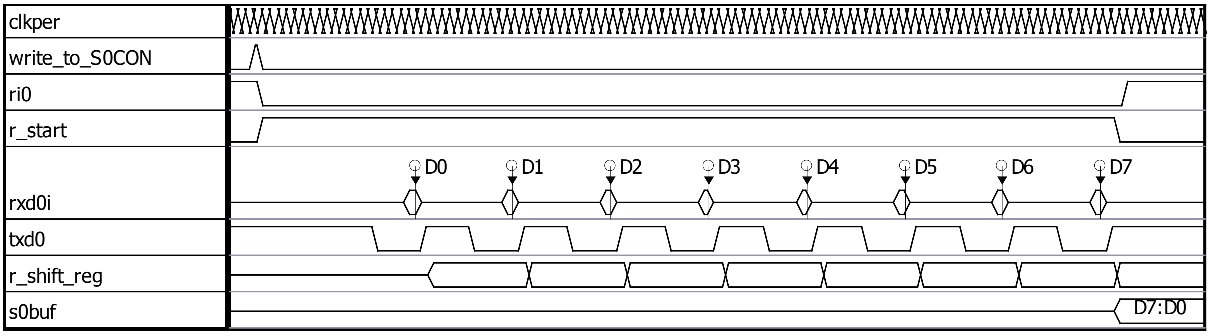


Figure 100. SERIAL0 Reception in Mode 0

- **Mode 1**

  In mode 1 the Serial Port 0 operates as asynchronous transmitter/receiver with 8 data bits and programmable baud rate. Additionally the baud rate can be doubled with the use of the "s0mod1" bit of the "pcon" register (2.4.29).

  Transmission is started by writing to the "s0buf" register. The "txd0" pin outputs data. The first bit transmitted is a start bit (always 0), then 8 bits of data proceed, after which a stop bit (always 1) is transmitted.

  The "rxd0i" pin inputs data. When reception starts, the Serial Port 0 synchronizes with the falling edge detected at pin "rxd0i". Input data are available after completion of the reception in the "s0buf" register, and the value of stop bit is available as the "rb80" flag in the "s0con" register. During the reception, the "s0buf" and "rb80" remain unchanged until the completion.

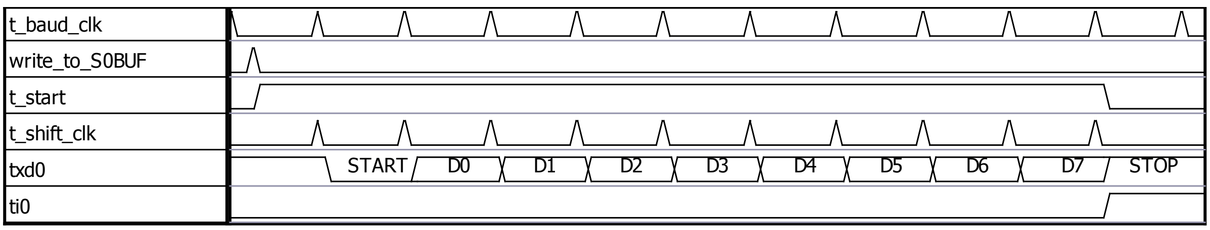


Figure 101. SERIAL0 Transmission in Mode 1

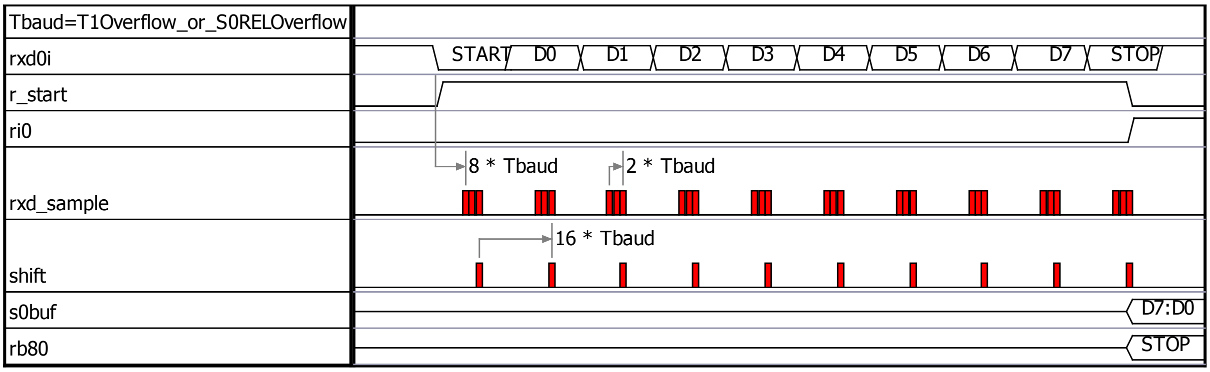


Figure 102. SERIAL0 Reception in Mode 1

- **Mode 2**

In mode 2 the Serial Port 0 operates as asynchronous transmitter/receiver with 9 data bits and baud rate fixed to Fclkper/32 or Fclkper/64, depending on the setting of "s0mod1" bit of "pcon" register (2.4.29).

Transmission is started by writing to the "s0buf" register. The "txd0" pin outputs data. The first bit transmitted is a start bit (always 0), then 9 bits of data proceed where the 9th is taken from bit "tb80" of the "s0con" register, after which a stop bit (always 1) is transmitted.

The "rxd0i" pin inputs data. When reception starts, the Serial Port 0 synchronizes with the falling edge detected at pin "rxd0". Input data are available after completion of the reception in the "s0buf" register, and the 9th bit is available as the "rb80" flag in the "s0con" register. During the reception, the "s0buf" and "rb80" remain unchanged until the completion.

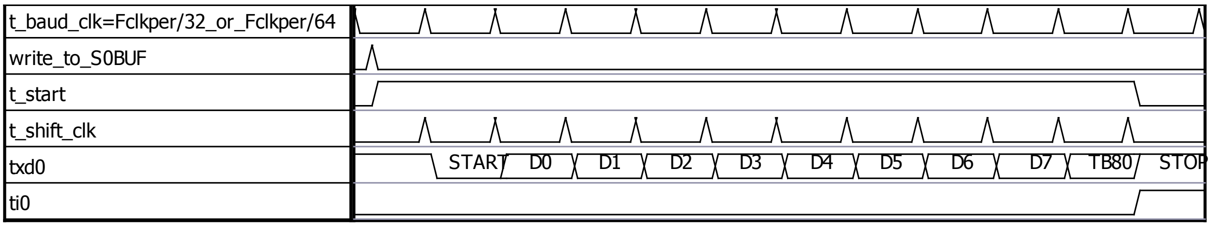


Figure 103. SERIAL0 Transmission in Mode 2

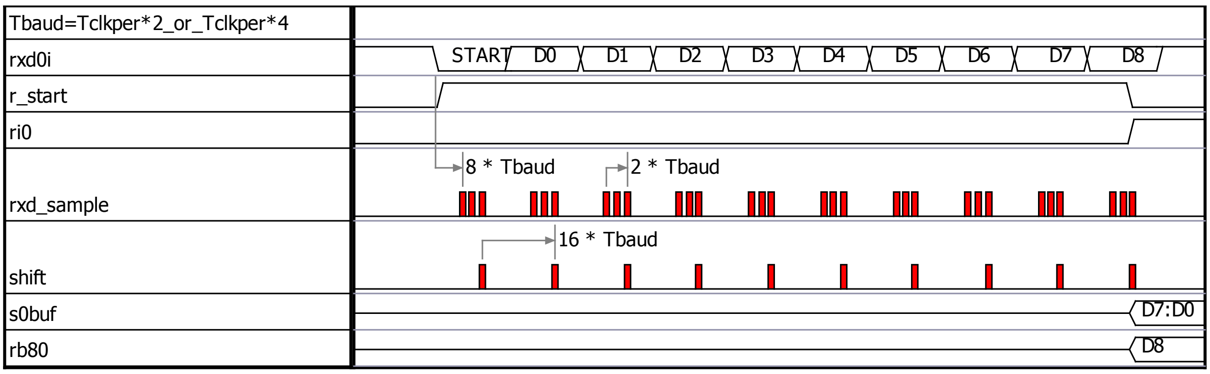


Figure 104. SERIAL0 Reception in Mode 2

- **Mode 3**

The only difference between Mode 2 and Mode 3 is that in Mode 3 either internal baud rate generator or Timer 1 can be used to specify the baud rate.

In mode 3 the Serial Port 0 operates as asynchronous transmitter/receiver with 9 data bits and programmable baud rate. Additionally the baud rate can be doubled with the use of the "s0mod7" bit of the "pcon" register (2.4.29).

Transmission is started by writing to the "s0buf" register. The "txd0" pin outputs data. The first bit transmitted is a start bit (always 0), then 9 bits of data proceed where the 9th is taken from bit "tb80" of the "s0con" register, after which a stop bit (always 1) is transmitted.

The "rxd0i" pin inputs data. When reception starts, the Serial Port 0 synchronizes with the falling edge detected at pin "rxd0". Input data are available after completion of the reception in the "s0buf" register, and the 9th bit is available as the "rb80" flag in the "s0con" register. During the reception, the "s0buf" and "rb80" remain unchanged until the completion.

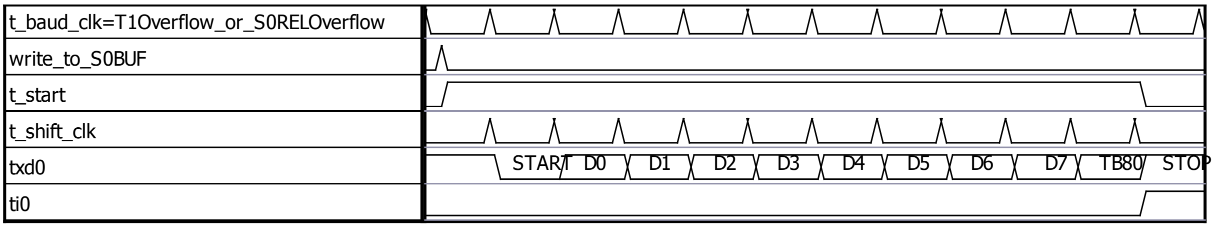


Figure 105. SERIAL0 Transmission in Mode 3

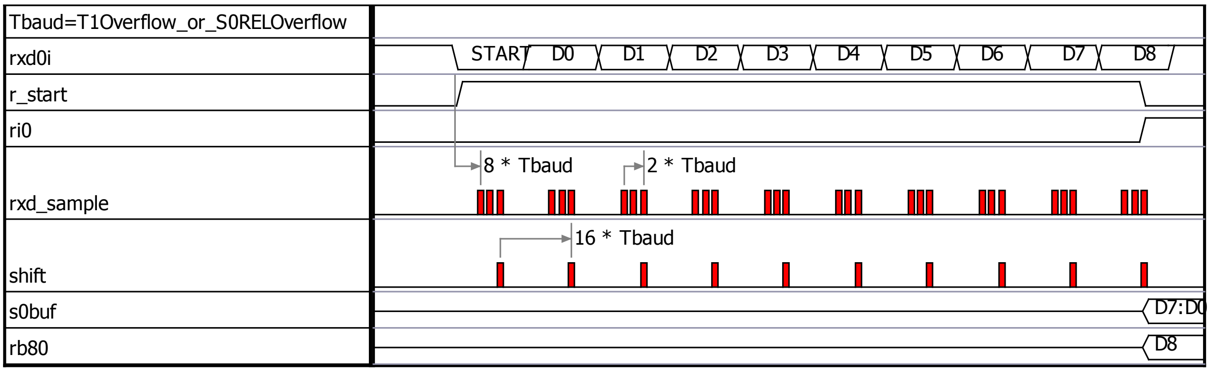


Figure 106. SERIAL0 Reception in Mode 3

c) The Serial Port 0 Multiprocessor Communication

The feature of receiving 9 bits in Modes 2 and 3 of Serial Interface 0 can be used for multiprocessor communication.

When the "sm20" bit of the "s0con" register (2.4.34) is set, the receive interrupt is generated only when the 9th received bit ("rb80" of "s0con") is 1. Otherwise, no interrupt is generated upon reception.

To utilize this feature to multiprocessor communication, the slave processors have their "sm20" bit set to 1. The master processor transmits the slave's address, with the 9th bit set to 1, causing reception interrupt in all of the slaves. The receiver compares the received byte with their slave address. If there is a match, the addressed slave clears its "sm20" flag and the rest of the message is transmitted from the master with the 9th bit set to 0. The other slaves keep their "sm20" set to 1 so that they ignore the rest of the message sent by the master.

d) Baud rates

In synchronous Mode 0 baud rate is fixed at Fclkper/12. Mode 2 has two optional baud rates (selected by the "s0mod1" bit in the "pcon" register). In modes 1 and 3 the baud rate is generated by overflow of Timer 1 and(or) Timer 2.

5.9. SERIAL1

5.9.1 Overview

The SERIAL1 provides a flexible full-duplex asynchronous receiver/transmitter. It can operate in two modes. The SERIAL1 is buffered at the receive side, i.e. it can reveive new data while the previously received is not damaged in the receive register until the completion of the 2nd transfer.

5.9.2 Pin Description

Table 95. SERIAL1 Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	High	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
newinstr	I	high	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
Serial Port Interface			
rxd1ff	I	-	Serial Port 1 receive data input
txd1	O	-	Serial Port 1 transmit data output
Interrupts			
ri1	O	High	Serial Port 1 receive flag
ti1	O	High	Serial Port 1 transmit flag
Special Function Register interface			
sfrdatai	I	8	data input bus
sfraddr	I	7	address bus
sfrwe	I	High	write enable
s1con	O	8	Serial 1 Control Register
s1buf	O	8	Serial 1 Data Buffer
s1rell	O	8	Serial 1 Baud Rate Generator Reload Register (low-order byte)
s1relh	O	8	Serial 1 Baud Rate Generator Reload Register (high-order byte)

5.9.3 Block Diagram

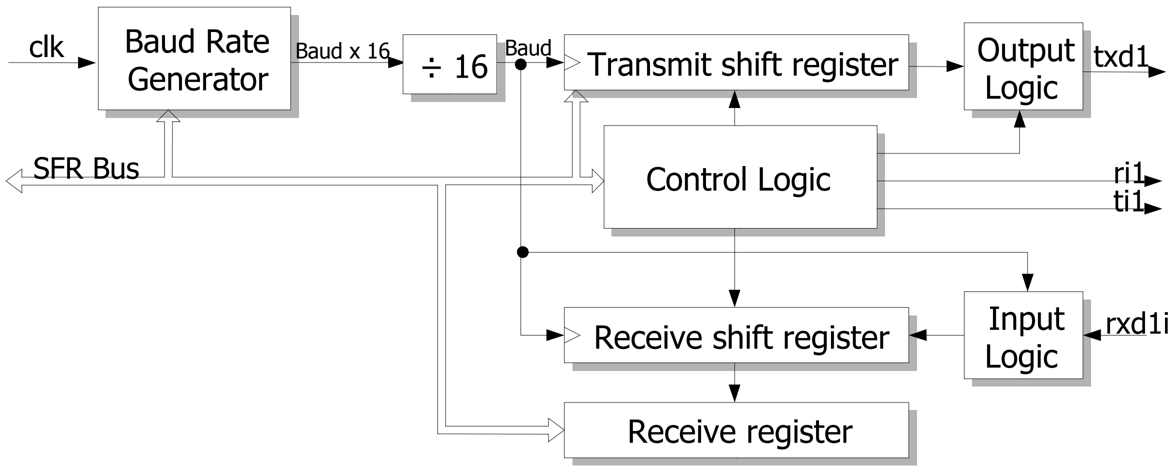


Figure 107. SERIAL1 block diagram

5.9.4 Description

a) The Serial Port 1 Baud Rate Generation

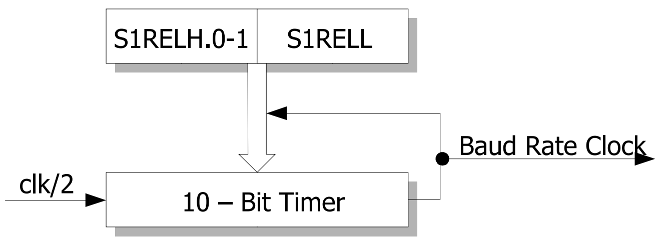


Figure 108. SERIAL1 Baud rate generation diagram

b) The Serial Port 1 Operating Modes

- **Mode A**

In mode A the Serial Port 1 operates as asynchronous transmitter/receiver with 9 data bits and programmable baud rate. The "s1relh", "s1rell" (2.4.40) baud rate generator is used to synchronize input and output transfers. The baud rate of Serial Port 1 cannot be modified using the "s0mod1" bit of the "pcon" register.

Transmission is started by writing to the "s1buf" register (2.4.38). The "txd1" pin outputs data. The first bit transmitted is a start bit (always 0), then 9 bits of data proceed where the 9^{th} is taken from bit "tb81" of the "s1con" register (2.4.39), after which a stop bit (always 1) is transmitted.

The "rxd1i" pin inputs data. When reception starts, the Serial Port 1 synchronizes with the falling edge detected at pin "rxd1i". Input data are available after completion of the reception in the "s1buf" register (2.4.38), and the 9^{th} bit is available as the "rb81" flag in the "s1con" register (2.4.39). During the reception, the "s1buf" and "rb81" remain unchanged until the completion.

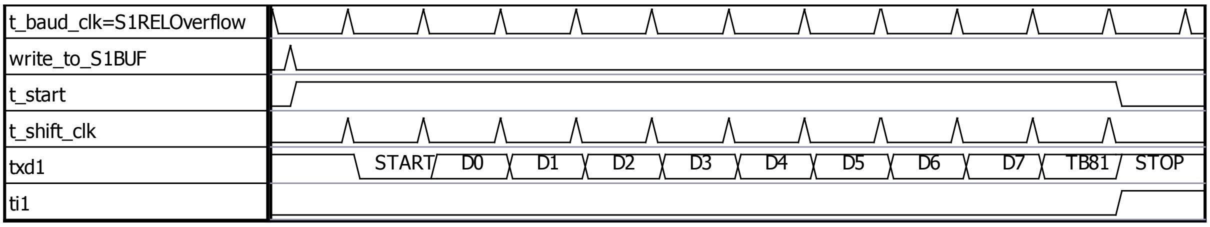


Figure 109. SERIAL1 Transmission in Mode A

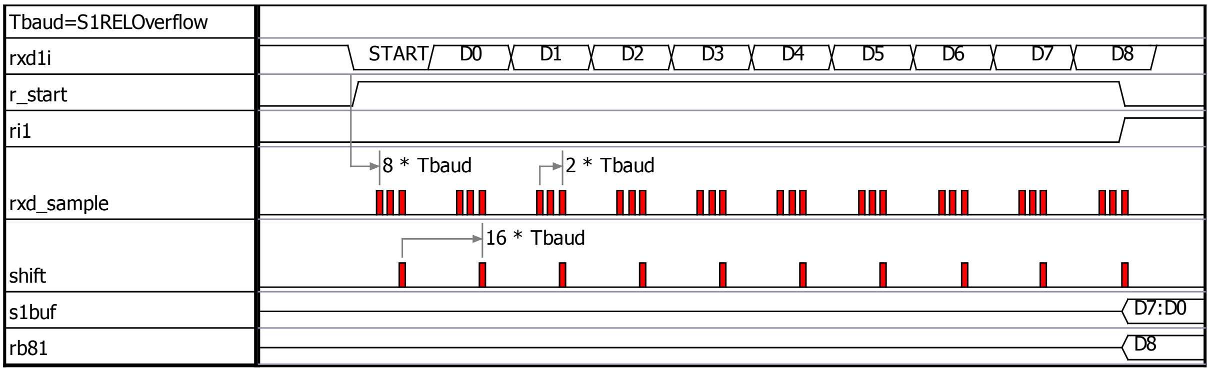


Figure 110. SERIAL1 Reception in Mode A

- **Mode B**

  In mode B the Serial Port 1 operates as asynchronous transmitter/receiver with 8 data bits and programmable baud rate. The "s1relh", "s1rell" (2.4.40) baud rate generator is used to synchronize input and output transfers. The baud rate cannot be mofified using the "s0mod1" bit of the "pcon" register.

  Transmission is started by writing to the "s1buf" register (2.4.38). The "txd1" pin outputs data. The first bit transmitted is a start bit (always 0), then 8 bits of data proceed, after which a stop bit (always 1) is transmitted.

  The "rxd1i" pin inputs data. When reception starts, the Serial Port 0 synchronizes with the falling edge detected at pin "rxd1i". Input data are available after completion of the reception in the "s1buf" register (2.4.38), and the value of stop bit is available as the "rb81" flag in the "s1con" register (2.4.39). During the reception, the "s1buf" and "rb81" remain unchanged until the completion.

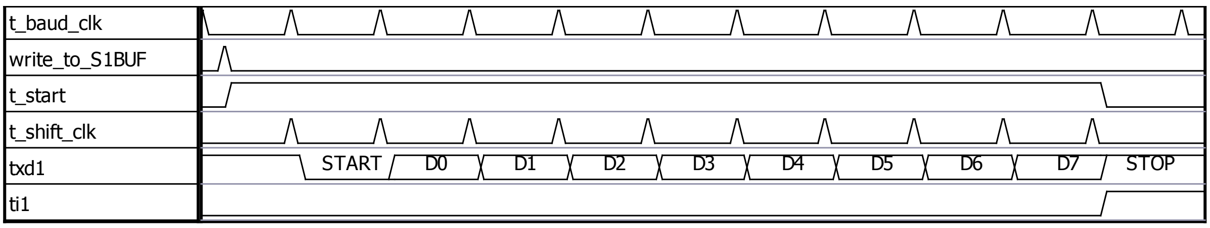


Figure 111. SERIAL1 Transmission in Mode B

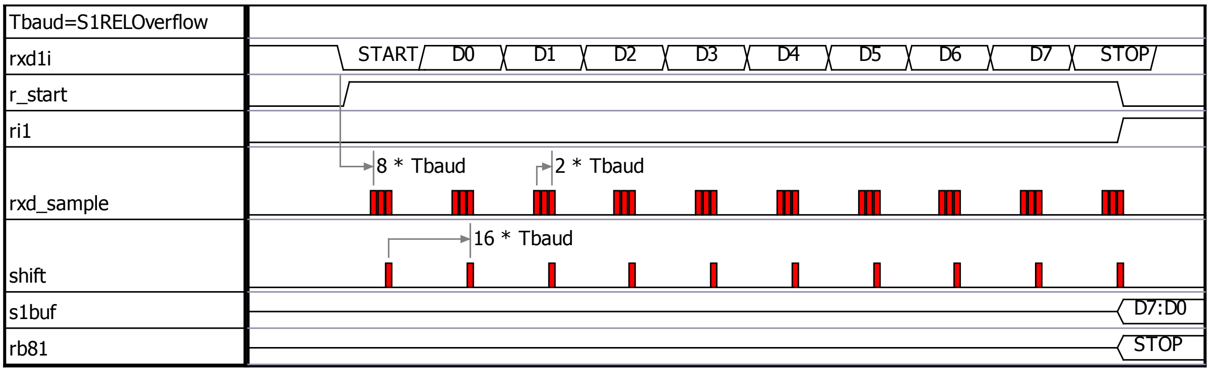


Figure 112. SERIAL1 Reception in Mode B

c) The Serial Port 1 Multiprocessor Communication

The feature of receiving 9 bits in Mode A of Serial Interface 1 can be used for multiprocessor communication.

When the "sm21" bit of the "s1con" register(2.4.39) is set, the receive interrupt is generated only when the 9th received bit ("rb81" of "s1con") is 1. Otherwise, no interrupt is generated upon reception.

To utilize this feature to multiprocessor communication, the slave processors have their "sm21" bit set to 1. The master processor transmits the slave's address, with the 9th bit set to 1, causing reception interrupt in all of the slaves. The slave processors' software compares the received byte with their network address. If there is a match, the addressed slave clears its "sm21" flag and the rest of the message is transmitted from the master with the 9th bit set to 0. The other slaves keep their "sm21" set to 1 so that they ignore the rest of the message sent by the master.

5.10. WATCHDOG

5.10.1 Overview

The watchdog timer is a 14-bit counter that is incremented every 12 clock cycles. It is used to provide the system supervision in case of software or hardware upset. If the software is not able to refresh the Watchdog Timer after 196608 clock cycles (16,384 ms when using 12MHz clock), an internal reset is generated.

5.10.2 Pin Description

Table 96. Watchdog Timer Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral Clock** Clock input for all internal synchronous logic
resetff	I	High	**Registered "reset" signal**, synchronized at the rising edge of "clkper", used to reset the Watchdog Timer and to detect the Start Input ("swd")
newinstr	I	high	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
wdts	O	High	Watchdog timer status flag output

Name	Type	Polarity Bus size	Description
Debug interface			
debugack	I	High	**Debug acknowledge** Indicates that the CPU has stopped instruction execution and entered the Debug Mode
watchdogstop	I	High	**Stop the Watchdog Timer** It is set by the CPU to syspend the Watchdog Timer when the CPU is in Debug Mode
Special Function Register interface			
sfrdatai	I	8	Input data bus
sfraddr	I	7	Address bus
sfrwe	I	High	Write enable

5.10.3 Description

The Watchdog Timer consists of 14-bit counter (not accessible as SFR) and control logic.

The Watchdog Timer is incremented every 12 clock cycles, which makes the whole period to be 12*16384=196608 clock cycles long.

a) Start Procedure

After hardware reset the Watchdog Timer is inactive. The only way to start the Watchdog Timer is by software. The Watchdog Timer is started by writing the two-byte sequence (write 1Eh and next E1h values to the "wdtrst" register.

When watchdog counter enters the state of 3FFFh, the internal reset is generated as the "wdts" output is active.

b) Refreshing the Watchdog Timer

The watchdog timer must be refreshed regularly to prevent reset request signal ("wdts") from becoming active. This requirement imposes obligation on the programmer to issue two followed instructions. The first instruction writes the 1Eh value to the "wdtrst" register and the second one writes the E1h value to the "wdtrst" register. The maximum allowed delay between writing proper values is 1 instruction cycles (that means the instructions which load both values to the "wdtrst" register are not separated with any other instruction). After that the internal watchdog counter is automatically cleared, which prevents the Watchdog Timer reset flag generating.

5.11. PMURSTCTRL

5.11.1 Overview

The main purpose of the **PMURSTCTRL** subcomponent is to provide the R80251XC with two operation modes with limited power consumption (IDLE or STOP) and to generate the internal synchronous reset signal.

The power down modes are implemented as two outputs controlling the behaviour of the off-core clock generator. The **PMURSTCTRL** generates output signals named "clkcpuen" and "clkperen" which should be externally used as gating control signals for clocks connected with appropriate R80251XC inputs ("clkcpu" and "clkper", respectively). These signals serve as clock enable signals for R80251XC CPU and peripherals.

The most of all registers and flip-flops in the R80251XC design are synchronously reset by high active internal "rst" signal. External hardware reset or watchdog timer reset can activate this signal. High level on external "reset" pin or watchdog reset request for at least two clock cycles while the "clk" is running resets the device. The "rst" signal is synchronized with the "clkper" in the **SYNCNEG** module, then it is combined with the other reset requests from Watchdog, Software Reset module and the OCDS in the **PMURSTCTRL**, and again it is synchronized by one flip-flop in the **SYNCNEG** module.

5.11.2 Pin Description

Table 97. PMURSTCTRL Pin Description

Name	Type	Polarity Bus size	Description
resetff	I	High	**Hardware reset input sample** This signal is sampled and used to generate the main device reset
Internal reset signals			
wdts	I	High	**Watchdog Timer status flag** Reset Request from Watchdog Timer
srst	I		**Software reset rquest** Reset request from software reset control logic.
rst	O	High	**Main device synchronous reset output** Combined reset signal derived from "reset" input, Watchdog Timer overflow, Software Reset and OCDS reset to all other subcomponents of the R80251XC (except Watchdog Timer and Software Reset components)
rsttowdt	O	High	**Watchdog synchronous reset output** Combined reset signal derived from "reset" input Software Reset and OCDS reset, to be synchronized in the SYNCNEG module, then connected to the Watchdog Timer component
rsttosrst	O	High	**Software Reset component synchronous reset output** Combined reset signal derived from "reset" input, Watchdog Timer overflow and OCDS reset, to be synchronized in the SYNCNEG module, then connected to the Software Reset component
PMU signals			
idle	I	High	Idle Mode Request from the CPU

Name	Type	Polarity Bus size	Description
stop	I	High	Stop Mode Request from the CPU
pmuintreg	I	High	Interrupt request from WAKEUPCTRL unit
clkcpu_en	O	High	**CPU clock enable output** Controls the off-core gated "clkcpu" signal, when disabled stops the clock
clkper_en	O	High	**Peripheral clock enable output** Controls the off-core gated "clkper" signal, when disabled stops the clock
cpu_resume	O	High	Used to wake-up the CPU after exitting from Power-Down Mode
hold	I	High	HOLD mode request input
holdacpu	I	High	HOLD acknowledge generated by CPU
holdapmu	O	High	HOLD acknowledge generated by PMURSTCTRL in Power-Down Mode
Debug signals			
debugrst	I	High	**Debug reset rquest** Reset request from debugger.
pmudreq	I	High	**Debug request input** Debug request signal (sychronized)
debug-pmureq	I	High	**Debug request input** Debug request signal (from other clock domain)
debugack	I	High	**Debug acknowledge** Indicates that the CPU has stopped instruction execution and entered the Debug Mode
debugperen	I	High	**Peripheral clock control input** Debugger signal – enable or disable peripheral clock.
debugpmu-req_synch	O	High	**Debug request ouput** Debug request signal (sychronized)

The Power Management Unit provides two power management modes: IDLE and STOP.

a) Idle Mode

Setting the "idle" bit of "pcon" register (2.4.29) invokes the IDLE mode. in the IDLE mode clock for peripherals is running (the "clkper_en"=1, "clkcpu_en"=0). Power consumption drops because the CPU clock is stopped. The CPU can exit the IDLE state with any interrupt or reset.

b) Stop Mode

Setting the "stop" bit of "pcon" register (2.4.29) invokes the STOP mode. Both clocks for the CPU and peripherals are stopped in this mode (the "clkper_en"=0, "clkcpu_en"=0). The CPU can exit this state with an external interrupt ("int0" or "int1") or reset. Internally generated interrupts (timer, serial port, ...) cannot be used since they require clock activity to operate.

The "clkcpu_en" output is dedicated to control the off-core gate on to the "clkcpu" input.

The "clkper_en" output is dedicated to control the off-core gate on to the "clkper" input.

c) Reset Generation

The extenrnal "reset" input is sampled by two rising-edge triggered flip-flops in the SYNCNEG module. The main device synchronous reset "rst" is activated three clock periods after hardware "reset" is active. The "rst" is active for minimum 2 clock cycles to allow the proper reset when Power Down mode is invoked.

When the Watchdog Timer overflow occurs, the "rst" is activated immediately.

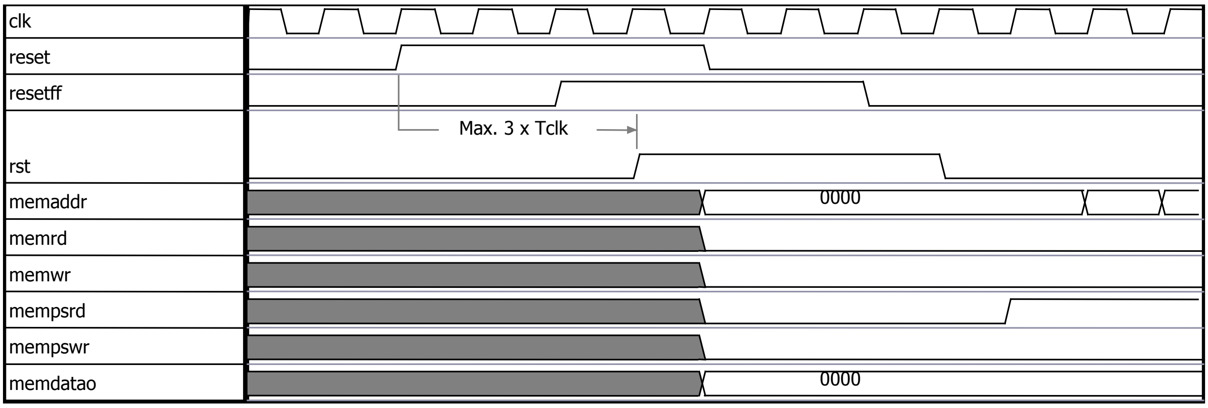


Figure 113. External Reset Timing

Note :

- clk – external clock input
- Tclk – clock period
- reset – external reset input
- resetff – external reset input sampled by 2 rising-edge triggered flip-flops
- rst – internally generated reset

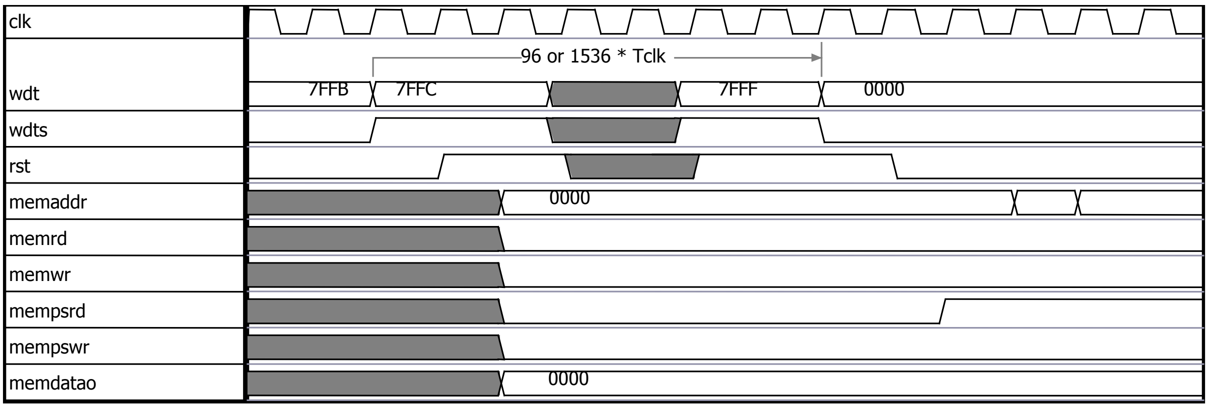


Figure 114. Watchdog Reset Timing

Note :

- clk – external clock input
- Tclk – clock period
- wdt – watchdog timer registers ("wdth", "wdtl")
- wdts – watchdog timer status flag
- rst – internally generated reset signal

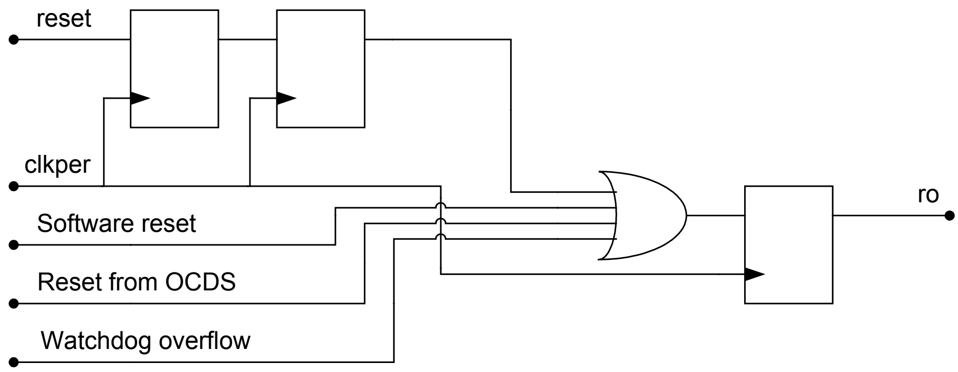


Figure 115. Reset Output Generation

d) Hold Mode

Normally the HOLD mode is serviced by the CPU. The exception is the Power-Down Mode. When IDLE or STOP mode is invoked, and the "hold" signal becomes active, the "holdapmu" is activated immediately and interrupts are suspended (no wake-up is possible unless "hold" is inactive again).

The "holdapmu" signal is combinationally OR-ed with the "holdacpu", generated synchronously by the CPU when not in STOP or IDLE mode. The "holda" output of the core is, depending on PMU_IMPLEMENT parameter, connected to "holdapmu" or "holdacpu".

5.12. PORTS

5.12.1 Overview

The **PORTS** subcomponent is composed of four 8-bit registers that can be accessed as Special Function Registers: "p0", "p1", "p2" and "p3". The contents of these registers is reflected at the output ports: "port0o", "port1o", "port2o" and "port3o", respectively.

To ensure the 8051-like PORTS, it is required for the corresponding inputs and outputs to be externally (off-core) connected using open-drain buffers.

5.12.2 Pin Description

Table 98. PORTS Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral Clock** Clock input for all internal synchronous logic
rst	I	High	**Synchronous reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
PORTS outputs			
port0o	O	8	Port 0 output
port1o	O	8	Port 1 output
port2o	O	8	Port 2 output
port3o	O	8	Port 3 output
Special Function Register interface			

Name	Type	Polarity Bus size	Description
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"

5.12.3 Block Diagram

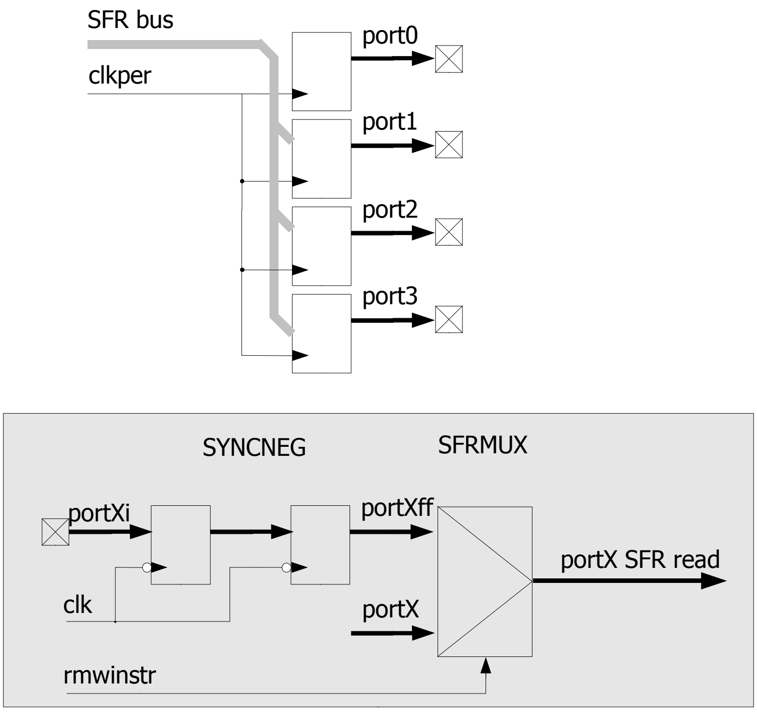


Figure 116. PORTS Block Diagram

5.12.4 Description

The **PORTS** subcomponent consists of four 8-bit registers "p0", "p1", "p2" and "p3" (2.4.28) which can be accessed as Special Function Registers while performing write or Read-Modify-Write instructions.

The contents of "p0", "p1", "p3" and "p3" registers can also be observed at the corresponding output ports of the core.

The related inputs of the "p0"..."p3" are sampled by the SYNCNEG module to filter-out the metastable states. The Read-Modify-Write access to the PORTS is performed in the SFRMUX module, where either the contents of an output register ("port0".."port3") or the sampled inputs ("port0ff".."port3ff") are selected using the "rmwinstr" signal.

5.13. WAKEUPCTRL

5.13.1 Overview

The purpose of the WAKEUPCTRL is to calculate interrupt request signal when the processor is in the STOP mode. The CPU and peripheral clocks are stopped in that mode, therefore the interrupt request for the PMURSTCTRL has to be derived combinationally. If "int0" or "int1" conditions are met, the "pmuintreq" output is activated and it enables back the clock activity in the PMURSTCTRL component. The "pmuintreq" output can also be set when the processor is in the IDLE mode, when the peripheral clock is running and the interrupt controller (ISR) generates an interrupt request.

5.13.2 Pin Description

Table 99. WAKEUPCTRL Pin Description

Name	Type	Polarity Bus size	Description
irq	I	High	Interrupt request from ISR
int0ff	I	Low/Fall	External interrupt 0
int1ff	I	Low/Fall	External interrupt 1
it0	I	-	External Interrupt 0 type select
it1	I	-	External Interrupt 1 type select
isreg	I	4	**In-service register** Contains the priority level of interrupt in progress
intprior0	I	2	External Interrupt 0 priority
intprior1	I	2	External Interrupt 1 priority
eal	I	High	Global interrupt enable
eint0	I	High	Interrupt 0 enable
eint1	I	High	Interrupt 1 enable
pmuintreq	O	High	Interrupt request to PMURSTCTRL

5.13.3 Block Diagram

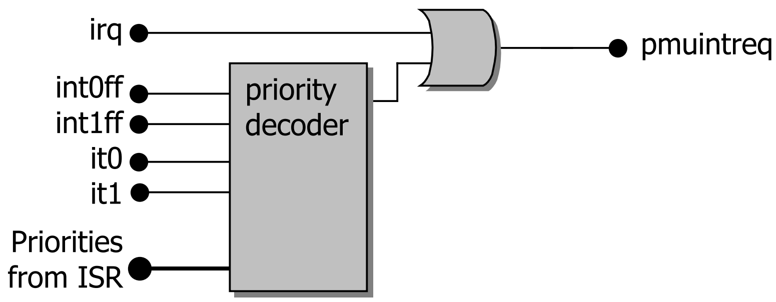


Figure 117. WAKEUPCTRL Block Diagram

5.13.4 Description

When the IDLE mode is invoked, the ISR and other peripherals are clocked normally and interrupts are generated normally, therefore the "irq" signal coming from the ISR module can directly activate the "pmuintreq" output to finish the IDLE mode.

When the STOP mode is invoked, neither the "clkcpu" nor "clkper" are working. The ISR module can't generate an interrupt since no peripherals are working. The only interrupts that may be accepted in the STOP mode are External Interrupt 0 and 1 ("int0", "int1"). The additional interrupt priority decoder (combinational) which is introduced in the WAKEUPCTRL detects whether External Interrupt 0 or 1 is enabled and at proper priority level so that it can be accepted. If so, the "pmuintreq" output is activated, causing the PMURSTCTRL to exit from the STOP mode.

5.14. SFRMUX

5.14.1 Overview

The SFRMUX module serves a global Special Function Registers bus multiplexer. When any of the internal or external SFRs is being read with the use of direct addressing, that data is passed through the SFRMUX to the CPU.

5.14.2 Pin Description

Table 100. SFRMUX Pin Description

Name	Type	Polarity Bus size	Description
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
c	I	High	**Carry flag** Carry bit in arithmetic operations and accumulator for Boolean operations
ac	I	High	**Auxiliary Carry flag** Set if there is a carry-out from 3rd bit of Accumulator in BCD operations

Name	Type	Polarity Bus size	Description
f0	I	-	**General purpose Flag 0** General purpose flag available for user
rs	I	2	**Register bank select** bus, used to select working register bank
ov	I	High	**Overflow flag** Set in case of overflow in Accumulator during arithmetic operations
ud	I	-	**User defined flag (General purpose Flag 1)** General purpose flag available for user
p	I	High	**Parity flag** Reflects the parity of the Accumulator contents
n	I	High	**Negative flag** This bit is set if the result of the last logical or arithmetic operation was negative (i.e., bit 15 = 1). Otherwise it is cleared.
z	I	High	**Zero flag** This flag is set if the result of the last logical or arithmetic operation is zero. Otherwise it is cleared.
acc	I	8	**Accumulator register**
b	I	8	**B register**
dpl	I	8	**Data Pointer Low register** Contains bits 7:0 of the Data Pointer Register
dph	I	8	**Data Pointer High register** Contains bits 15:8 of the Data Pointer Register
dpxl	I	8	**Data Pointer Extended Low register** DPXL is the lower byte of the upper word of the extended data pointer, DPX = DR56
uconfig0_65	I	2	**User Config register 0 (wsa)** uconfig0.6, uconfig0.5 The number of wait states for external memory accesses. (all regions except 01:)
uconfig0_0	I	1	**User Config register 0 (src)** uconfig0.0 Source Mode/Binary Mode Select
uconfig1_4	I	1	**User Config register 1 (intr)** uconfig1.4 Interrupt Mode
uconfig1_21	I	2	**User Config register 1 (wsb)** uconfig1.2, uconfig1.1 The number of wait states for external memory accesses (Region 01:).
sp	I	8	**Stack Pointer** Points to the Top-of-stack
sph	I	8	**Stack Pointer high** SPH is the upper byte of the lower word of DR60, the extended stack pointer (SPX).

Name	Type	Polarity Bus size	Description
s0mod	I	2	**Serial Port 0: "baud rate doubler" and "s0con.7 select".**
gf0	I	-	**General purpose bit** pcon.2
p2sel	I	High	**High-order address byte configuration bit** pcon.3
stop	I	High	**Stop mode control** Setting this bit activates the Stop Mode. This bit is always read as 0
idle	I	High	**Idle mode control** Setting this bit activates the Idle Mode. This bit is always read as 0
port0	I	8	**Parallel Port 0 Output Register**
port1	I	8	**Parallel Port 1 Output Register**
port2	I	8	**Parallel Port 2 Output Register**
port3	I	8	**Parallel Port 3 Output Register**
port0ff	I	8	**Parallel Port 0 Input Register**
port1ff	I	8	**Parallel Port 1 Input Register**
port2ff	I	8	**Parallel Port 2 Input Register**
port3ff	I	8	**Parallel Port 3 Input Register**
rmwinstr	I	High	**Read-Modify-Write Instruction Indicator** Used to select between Port Output / Port Input register for read
t0_tmod	I	4	**Timer 0 Mode register** tmod.0-tmod.3 bits
t0_tf0	I	High	**Timer 0 overflow flag**
t0_tf1	I	High	**Timer 1 overflow flag** Generated by Timer 0 in Mode 3
t0_tr0	I	High	**Timer 0 run control flag**
tl0	I	8	**Timer 0 Low Register**
th0	I	8	**Timer 0 High Register**
t1_tmod	I	4	**Timer 1 Mode register** tmod.4-tmod.7 bits
t1_tf1	I	High	**Timer 1 overflow flag**
t1_tr1	I	High	**Timer 1 run control flag**
tl1	I	8	**Timer 1 Low Register**
th1	I	8	**Timer 1 High Register**
wdtrel	I	8	**Watchdog Timer Reload Register**
ip0wdts	I	High	**Watchdog Timer Status Flag** Bit ip0.6
wdt_tm	I	High	**Watchdog Timer Test Mode**
tl2	I	8	**Timer 2 Low Register**
th2	I	8	**Timer 2 High Register**
exen2	I	High	**External interrupt 2 enable flag**

Name	Type	Polarity Bus size	Description
crcl	I	8	**Compare/Capture Register 0 (low-order byte)**
crch	I	8	**Compare/Capture Register 0 (high-order byte)**
t2con	I	8	**Timer 2 Control Register**
ccen	I	8	**Compare/Capture Enable Register**
ccl1	I	8	**Compare/Capture Register 1 (low-order byte)**
cch1	I	8	**Compare/Capture Register 1 (high-order byte)**
ccl2	I	8	**Compare/Capture Register 2 (low-order byte)**
cch2	I	8	**Compare/Capture Register 2 (high-order byte)**
ccl3	I	8	**Compare/Capture Register 3 (low-order byte)**
cch3	I	8	**Compare/Capture Register 3 (high-order byte)**
tf2	I	High	**Timer 2 overflow flag**
exf2	I	High	**Timer 2 external flag**
s0con	I	8	**Serial Port 0 Control Register**
s0buf	I	8	**Serial Port 0 Data Buffer**
s0rell	I	8	**Serial Port 0 Reload Register (low-order byte)**
s0relh	I	8	**Serial Port 0 Reload Register (high-order byte)**
	I	High	**Serial Port 0 Baud Rate Select flag** When set, an additional baud rate generator is used for Serial Port 0, otherwise Timer 0 overflow is used
s1con	I	8	**Serial Port 1 Control Register**
s1buf	I	8	**Serial Port 1 Data Buffer**
s1rell	I	8	**Serial Port 1 Reload Register (low-order byte)**
s1relh	I	8	**Serial Port 1 Reload Register (high-order byte)**
ie0	I	High	**External interrupt 0 flag** Bit tcon.1
it0	I	High	**External interrupt 0 type control** Bit tcon.0
ie1	I	High	**External interrupt 1 flag** Bit tcon.3
it1	I	High	**External interrupt 1 type control** Bit tcon.2
iex2	I	High	**External interrupt 2 edge flag**
iex3	I	High	**External interrupt 3 edge flag**
iex4	I	High	**External interrupt 4 edge flag**
iex5	I	High	**External interrupt 5 edge flag**
iex6	I	High	**External interrupt 6 edge flag**
iex7	I	High	**External interrupt 7 edge flag**
iex8	I	High	**External interrupt 8 edge flag**
iex9	I	High	**External interrupt 9 edge flag**
iex10	I	High	**External interrupt 10 edge flag**
iex11	I	High	**External interrupt 11 edge flag**
iex12	I	High	**External interrupt 12 edge flag**

Name	Type	Polarity Bus size	Description
int_req_ch0	I	High	**DMA Channel 0 interrupt flag**
int_req_ch1	I	High	**DMA Channel 1 interrupt flag**
int_req_ch2	I	High	**DMA Channel 2 interrupt flag**
int_req_ch3	I	High	**DMA Channel 3 interrupt flag**
int_req_ch4	I	High	**DMA Channel 4 interrupt flag**
int_req_ch5	I	High	**DMA Channel 5 interrupt flag**
int_req_ch6	I	High	**DMA Channel 6 interrupt flag**
int_req_ch7	I	High	**DMA Channel 7 interrupt flag**
dmas0	I	8	**DMA Source Address Register 0**
dmas1	I	8	**DMA Source Address Register 1**
dmas2	I	7	**DMA Source Address Register 2**
dmat0	I	8	**DMA Target Address Register 0**
dmat1	I	8	**DMA Target Address Register 1**
dmat2	I	7	**DMA Target Address Register 2**
dmac0	I	8	**DMA Byte Counter Register 0**
dmac1	I	8	**DMA Byte Counter Register 1**
dmac2	I	7	**DMA Byte Counter Register 2**
dmasel	I	8	**DMA Select Register**
dmam0	I	8	**DMA Mode Register 0**
dmam1	I	8	**DMA Mode Register 1**
ien0	I	8	**Interrupt Enable 0 Register**
ien1	I	6	**Interrupt Enable 1 Register**
ien2	I	6	**Interrupt Enable 2 Register**
ip0	I	6	**Interrupt Priority 0 Register**
ip1	I	6	**Interrupt Priority 1 Register**
isr_testmode	I	High	**Interrupt Controller Test Mode Flag**
i2c_int	I	High	**I2C interrupt request signal**
i2cdat_o	I	8	**I2C Data Register**
i2cadr_o	I	8	**I2C Own Slave Address Register**
i2ccon_o	I	8	**I2C Control Register**
i2csta_o	I	8	**I2C Status Data Register**
i2c2_int	I	High	**Secondary I2C interrupt request signal**
i2c2dat_o	I	8	**Secondary I2C Data Register**
i2c2adr_o	I	8	**Secondary I2C Own Slave Address Register**
i2c2con_o	I	8	**Secondary I2C Control Register**
i2c2sta_o	I	8	**Secondary I2C Status Data Register**
intspi	I	High	**Serial Peripheral Interface interrupt request signal**
spcon	I	8	**Serial Peripheral Interface Control Register**
spsta	I	8	**Serial Peripheral Interface Status Register**
spdat	I	8	**Serial Peripheral Interface Data Register**
spssn	I	8	**Serial Peripheral Interface Slave Select Register**

Name	Type	Polarity Bus size	Description
sfrdatai	I	8	**External SFR Bus input**
srstflag	I	High	**Software reset flag**
tf1_gate	O	High	**Timer 1 interrupt request signal** Set when either t0_tf1 or t1_tf1 is active
riti0_gate	O	High	**Serial Port 0 Receive/Transmit Interrupt request signal** Set when either the "ri0" or "ti0" flag of the "s0con" register is active
riti1_gate	O	High	**Serial Port 1 Receive/Transmit Interrupt request signal** Set when either the "ri1" or "ti1" flag of the "s1con" register is active
tfexf2_gate	O	High	**Timer 2 Interrupt request signal** Set when either the "tf2" or "exf2" flag of the "ircon" register is active
iex2_gate	O	High	**External Interrupt 2 / SPI interrupt request signal** Set when either the "iex2" of the "ircon" register or the "intspi" (SPI interrupt request) is active
iex7_gate	O	High	**External Interrupt 7 / I2C interrupt request signal** Set when either the "iex7" of the "ircon" register or the "i2c_int" (I2C interrupt request) is active
iex8_gate	O	High	**External Interrupt 8 / DMA Channels 0 & 4 interrupt** Set when either the "iex8" of the "ircon2" register or the "dmaint0" or "dmaint4" (DMA Channel 0 & 4 interrupt request) is active
iex9_gate	O	High	**External Interrupt 9 / DMA Channels 1 & 5 interrupt** Set when either the "iex9" of the "ircon2" register or the "dmaint1" or "dmaint5" (DMA Channel 1 & 5 interrupt request) is active
iex10_gate	O	High	**External Interrupt 10 / DMA Channels 2 & 6 interrupt** Set when either the "iex10" of the "ircon2" register or the "dmaint2" or "dmaint6" (DMA Channel 2 & 6 interrupt request) is active
iex11_gate	O	High	**External Interrupt 11 / DMA Channels 4 & 7 interrupt** Set when either the "iex11" of the "ircon2" register or the "dmaint3" or "dmaint7" (DMA Channel 3 & 7 interrupt request) is active
sfrdatao	O	8	**SFR data bus output** Output of the SFR multiplexer
int_ack_03_isr int_ack_03_dma int_ack_03	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_0b_isr int_ack_0b_dma int_ack_0b	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_13_isr int_ack_13_dma int_ack_13	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**

Name	Type	Polarity Bus size	Description
int_ack_1b_isr int_ack_1b_dma int_ack_1b	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_43_isr int_ack_43_dma int_ack_43	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_4b_isr int_ack_4b_dma int_ack_4b	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_53_isr int_ack_53_dma int_ack_53	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_5b_isr int_ack_5b_dma int_ack_5b	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_63_isr int_ack_63_dma int_ack_63	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_6b_isr int_ack_6b_dma int_ack_6b	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_8b_isr int_ack_8b_dma int_ack_8b	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_93_isr int_ack_93_dma int_ack_93	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_9b_isr int_ack_9b_dma int_ack_9b	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_a3_isr int_ack_a3_dma int_ack_a3	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
int_ack_ab_isr int_ack_ab_dma int_ack_ab	I I O	High	**Interrupt acknowledge by ISR input** **Interrupt acknowledge by DMA input** **Combined Interrupt acknowledge output**
ext_sfr_sel	O	High	**External SFR detect output** Set active when an external SFR is accessed

5.14.3 Block Diagram

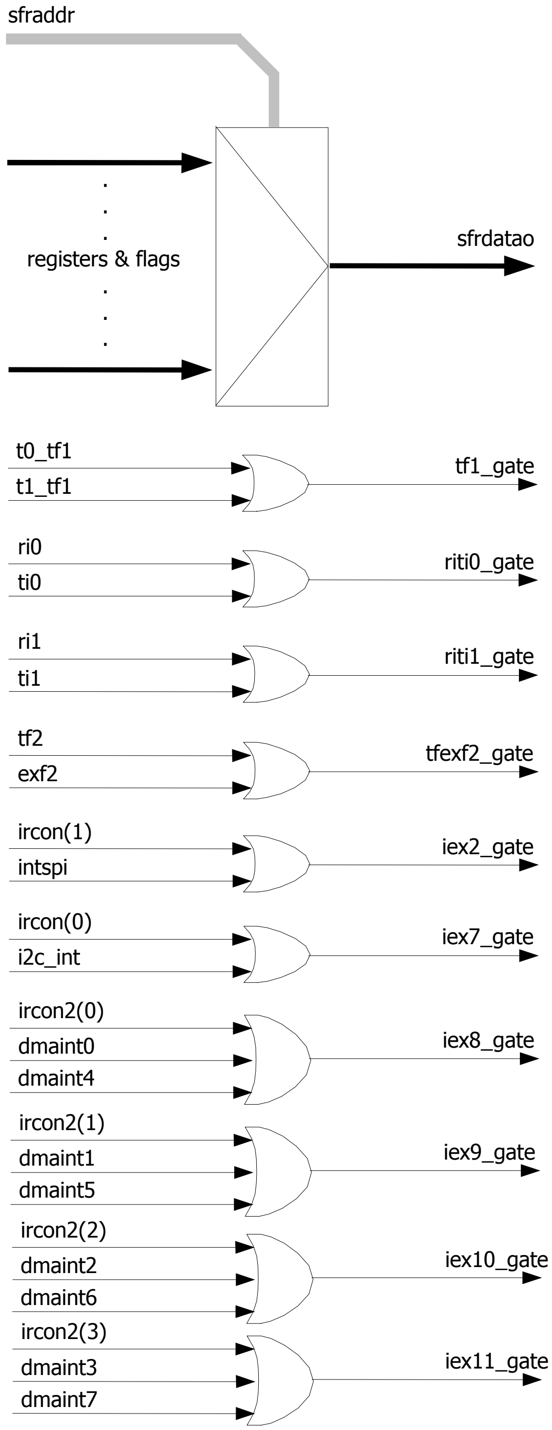


Figure 118. SFRMUX Block Diagram

5.14.4 Description

The SFRMUX is a general Special Function Register Bus multiplexer. It selects one of the input registers pointed by the “sfraddr” bus and puts its contents onto the output bus “sfrdatao”, which is fed to the CPU. There is the only one SFR bus multiplexer in the whole R80251XC design.

Several registers, e.g. The “ircon” or “tcon”, are internally concatenated from single, non-vector inputs (separated flags).

All the register locations and eventual bit-wise assignments are defined in Chapter 2.4 : Special Function Registers.

The Parallel Port 0 – 3 registers can be read from two different sources depending on the state of the "rmwinstr" input, regardless of the "sfraddr". When a Read-Modify-Write instruction is performed, the output register of the selected Port is used instead of the input Port. This is shown in Figure 116.

The other purpose of the SFRMUX module is to generate interrupt request signals from Timer 1, Timer 2, Serial Port 0, Serial Port 1, External Interrupts 2 & 7, DMA channels 0-7 interrupts, SPI interrupt and I2C interrupt. Those interrupts are generated here because they can be invoked by setting more than one flag. Their sources are simply OR-ed and then routed to the interrupt controller as one signal.

This sub-component contains only combinational logic.

5.15. SYNCREGS

5.15.1 Overview

The SYNCNEG module incorporates the clock domain synchronization flip-flops for all external asynchronous inputs.

5.15.2 Pin Description

Table 101. SYNCNEG Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral Clock** Peripheral clock, used here to capture the state of asynchronous input signals into flip-flops
reset	I	High	**Hardware reset input** Conected directly to the "reset" input of the core
int0	I	Low/Fall	External Interrupt 0 input
int1	I	Low/Fall	External Interrupt 1 input
int2	I	Fall/Rise	External Interrupt 2 input
int3	I	Fall/Rise	External Interrupt 3 input
int4	I	Rise	External Interrupt 4 input
int5	I	Rise	External Interrupt 5 input
int6	I	Rise	External Interrupt 6 input
int7	I	Rise	External Interrupt 7 input
int8	I	Rise	External Interrupt 8 input
int9	I	Rise	External Interrupt 9 input
int10	I	Rise	External Interrupt 10 input
int11	I	Rise	External Interrupt 11 input
int12	I	Rise	External Interrupt 12 input
port0i	I	8	Parallel Port 0 input
port1i	I	8	Parallel Port 1 input
port2i	I	8	Parallel Port 2 input
port3i	I	8	Parallel Port 3 input

Name	Type	Polarity Bus size	Description
swd	I	High	Start Watchdog Timer input
t0	I	Fall	Timer 0 external input
t1	I	Fall	Timer 1 external input
t2	I	Fall	Timer 2 external input
t2ex	I	Fall	Timer 2 capture trigger input
cc	I	4	Compare/Capture inputs
rxd0i	I	-	Serial Port 0 data input
rxd1	I	-	Serial Port 1 data input
sdai	I	-	I2C Interface data input
misoi	I	-	Synchronous Peripheral Interface data input
int0ff	O	Low/Fall	External Interrupt 0 synchronization flip-flop output
int1ff	O	Low/Fall	External Interrupt 1 synchronization flip-flop output
int2ff	O	Fall/Rise	External Interrupt 2 synchronization flip-flop output
int3ff	O	Fall/Rise	External Interrupt 3 synchronization flip-flop output
int4ff	O	Rise	External Interrupt 4 synchronization flip-flop output
int5ff	O	Rise	External Interrupt 5 synchronization flip-flop output
int6ff	O	Rise	External Interrupt 6 synchronization flip-flop output
int7ff	O	Rise	External Interrupt 7 synchronization flip-flop output
int8ff	O	Rise	External Interrupt 8 synchronization flip-flop output
int9ff	O	Rise	External Interrupt 9 synchronization flip-flop output
int10ff	O	Rise	External Interrupt 10 synchronization flip-flop output
int11ff	O	Rise	External Interrupt 11 synchronization flip-flop output
int12ff	O	Rise	External Interrupt 12 synchronization flip-flop output
port0ff	O	8	Parallel Port 0 synchronization register output
port1ff	O	8	Parallel Port 1 synchronization register output
port2ff	O	8	Parallel Port 2 synchronization register output
port3ff	O	8	Parallel Port 3 synchronization register output
t0ff	O	Fall	Timer 0 external input synchronization flip-flop
t1ff	O	Fall	Timer 1 external input synchronization flip-flop
swdff	O	High	Start Watchdog Timer input synchronization flip-flop
t2ff	O	Fall	Timer 2 external input synchronization flip-flop
t2exff	O	Fall	Timer 2 capture trigger synchronization flip-flop
ccff	O	4	Compare/Capture inputs synchronization register
rxd0ff	O	-	Serial Port 0 data synchronization flip-flop
rxd1ff	O	-	Serial Port 1 data synchronization flip-flop
sdaiff	O	-	I2C Interface data synchronization flip-flop (for verification purposes only)
misoiff	O	-	Synchronous Peripheral Interface data synchronization flip-flop (for verification purposes only)
resetff	O	High	Hardware reset synchronization flip-flop
rsttowdt	I	High	Reset for Watchdog Timer before being synchronized
rsttosrst	I	High	Reset for Software Reset module before being synchronized

Name	Type	Polarity Bus size	Description
rst	I	High	Reset for the remaining modules before being synchronized
rsttowdtff	O	High	Reset for Watchdog Timer
rsttosrstff	O	High	Reset for Software Reset module
rstff	O	High	Reset for the remaining modules

5.15.3 Block Diagram

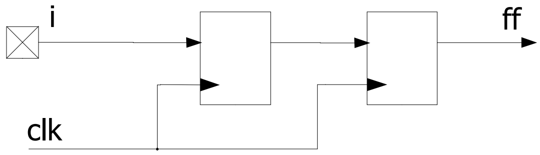


Figure 119. SYNCNEG Block Diagram

5.15.4 Description

The SYNCNEG module contains the synchronization flip-flops for all external asynchronous inputs. Each of the input signals is connected to a rising edge triggered flip-flop, which is then fed to another flip-flop, and then connected to the output of the SYNCNEG module. Then all of those signals are routed into various components of the R80251XC.

If it is assured that a signal from the list above is off-core synchronized to the system clock, the synchronization logic can be skipped for that signal, by modifying the code to substitute the clocked assignment with a direct assignment.

5.16. EXTINT

5.16.1 Overview

The EXTINT module contains edge-detection logic and request flags for all external interrupt inputs. Since the interrupt inputs are already synchronized with the system clock in the SYNCNEG module, therefore they are used together with only one flip-flop to detect rising or falling transitions.

5.16.2 Pin Description

Table 102. EXTINT Pin Description

Name	Type	Polarity Bus size	Description
clkper	I	Rise	**Peripheral clock** Clock input for all internal synchronous logic
rst	I	High	**Synchronous reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
newinstr	I	High	**New instruction indicator** Indicates the previous cycle was the first cycle of current instruction
int0ff	I	Fall/Low	**Externai Interrupt 0 input sample**

Name	Type	Polarity Bus size	Description
int0ack	I	High	**External Interrupt 0 acknowledge** Clears interrupt request flag
int1ff	I	Fall/Low	**Externai Interrupt 1 input sample**
int1ack	I	High	**External Interrupt 1 acknowledge** Clears interrupt request flag
int2ff	I	Fall/Rise	**Externai Interrupt 2 input sample**
iex2ack	I	High	**External Interrupt 2 acknowledge** Clears interrupt request flag
int3ff	I	Fall/Rise	**Externai Interrupt 3 input sample**
iex3ack	I	High	**External Interrupt 3 acknowledge** Clears interrupt request flag
int4ff	I	Rise	**Externai Interrupt 4 input sample**
iex4ack	I	High	**External Interrupt 4 acknowledge** Clears interrupt request flag
int5ff	I	Rise	**Externai Interrupt 5 input sample**
iex5ack	I	High	**External Interrupt 5 acknowledge** Clears interrupt request flag
int6ff	I	Rise	**Externai Interrupt 6 input sample**
iex6ack	I	High	**External Interrupt 6 acknowledge** Clears interrupt request flag
int7ff	I	Rise	**Externai Interrupt 7 input sample**
iex7ack	I	High	**External Interrupt 7 acknowledge** Clears interrupt request flag
int8ff	I	Rise	**Externai Interrupt 8 input sample**
iex8ack	I	High	**External Interrupt 8 acknowledge** Clears interrupt request flag
int9ff	I	Rise	**Externai Interrupt 9 input sample**
iex9ack	I	High	**External Interrupt 9 acknowledge** Clears interrupt request flag
int10ff	I	Rise	**Externai Interrupt 10 input sample**
iex10ack	I	High	**External Interrupt 10 acknowledge** Clears interrupt request flag
int11ff	I	Rise	**Externai Interrupt 11 input sample**
iex11ack	I	High	**External Interrupt 11 acknowledge** Clears interrupt request flag
int12ff	I	Rise	**Externai Interrupt 12 input sample**
iex12ack	I	High	**External Interrupt 12 acknowledge** Clears interrupt request flag
com	I	4	**Compare signals from Timer 2 / CCU**
ccen	I	8	**Compare / Capture Enable register from Timer 2 / CCU**
ie0	O	High	**External Interrupt 0 Edge flag**
it0	O	High	**External Interrupt 0 Type flag**

Name	Type	Polarity Bus size	Description
ie1	O	High	**External Interrupt 1 Edge flag**
it1	O	High	**External Interrupt 1 Type flag**
i2fr	O	High	**External Interrupt 2 Edge Select flag**
iex2	O	High	**External Interrupt 2 Edge flag**
i3fr	O	High	**External Interrupt 3 Edge Select flag**
iex3	O	High	**External Interrupt 3 Edge flag**
iex4	O	High	**External Interrupt 4 Edge flag**
iex5	O	High	**External Interrupt 5 Edge flag**
iex6	O	High	**External Interrupt 6 Edge flag**
iex7	O	High	**External Interrupt 7 Edge flag**
iex8	O	High	**External Interrupt 8 Edge flag**
iex9	O	High	**External Interrupt 9 Edge flag**
iex10	O	High	**External Interrupt 10 Edge flag**
iex11	O	High	**External Interrupt 11 Edge flag**
iex12	O	High	**External Interrupt 12 Edge flag**
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"

5.16.3 Block Diagram

All the elements of the EXTINT module are shown in their own block diagrams in paragraphs below (Figure 120 to Figure 132).

5.16.4 Description

a) External Interrupt 0

The External Interrupt 0 can be programmed to be low level active or falling transaction active. When interrupt condition is met, the "ie0" bit in "tcon" SFR (0) is set. Setting this bit forces interrupt request generation (if not masked), and corresponding interrupt vector is set. During the interrupt acknowledge cycle (when the CPU sets "intack" signal), when interrupt vector points to subroutine intended for external interrupt 0, the corresponding bit ("ie0") is automatically cleared by hardware, but only in case when this interrupt was set to be negative transaction sensitive. in case that external interrupt was set to be low level active, the interrupt subroutine must force external device to release the interrupt pin. Low level or negative transaction sensitivity is defined by "it0" flag in "tcon" (0) SFR.

The External interrupt 0 can be also triggered by setting a corresponding request bit of "tcon" by software.

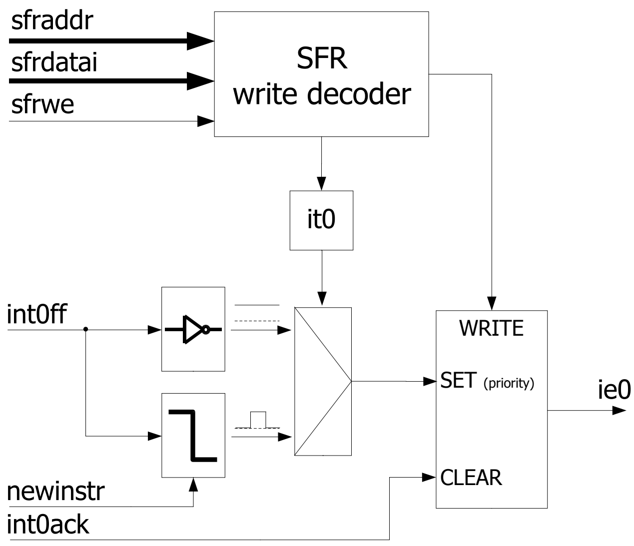


Figure 120. External Interrupt 0 Detection

b) External Interrupt 1

The External Interrupt 1 can be programmed to be low level active or falling transaction active. When interrupt condition is met, the "ie1" bit in "tcon" SFR (0) is set. Setting this bit forces interrupt request generation (if not masked), and corresponding interrupt vector is set. During the interrupt acknowledge cycle (when the CPU sets "intack" signal), when interrupt vector points to subroutine intended for external interrupt 1, the corresponding bit ("ie1") is automatically cleared by hardware, but only in case when this interrupt was set to be negative transaction sensitive. in case that external interrupt was set to be low level active, the interrupt subroutine must force external device to release the interrupt pin. Low level or negative transaction sensitivity is defined by "it1" flag in "tcon" (0) SFR.

The External interrupt 1 can be also triggered by setting a corresponding request bit of "tcon" by software.

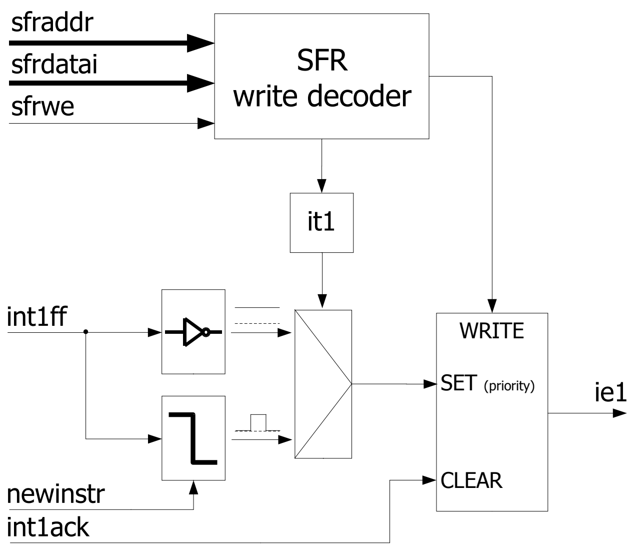


Figure 121. External Interrupt 1 Detection

c) External Interrupt 2

The External Interrupt 2 can be programmed to be positive or negative edge sensitive (depending on "i2fr" bit in "t2con" (2.4.47) SFR). When the chosen edge occurs, bit "iex2" in

"ircon" (2.4.26) SFR register is set and corresponding interrupt is invoked. Interrupt request flag "iex2" in "ircon" is cleared by hardware automatically when the service routine is vectored to.

The edge detection on "int2ff" input and setting the "iex2" flag in "ircon" SFR, in hardware, is made according to the scheme shown on Figure 122.

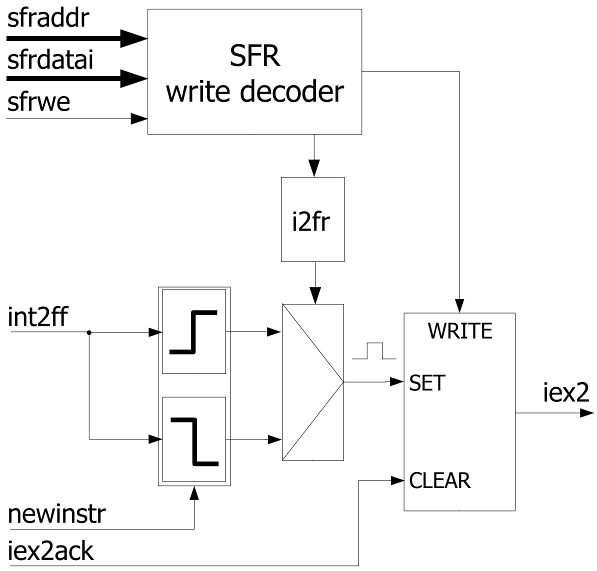


Figure 122. External Interrupt 2 Detection

The External Interrupt 2 (after being synchronized to the system clock in the SYNCNEG module) is checked for rising or falling transition, depending on the setting of the "i2fr" flag, which is also implemented in EXTINT module. The "iex2" interrupt request flag can be set by hardware upon detection of chosen edge of the interrupt signal, and can be cleared by hardware upon receiving acknowledge signal from the interrupt controller (ISR module).

The interrupt edge detection module is protected against losing interrupt requests which occur during a Read-Modify-Write instructions performed on the interrupt request register. The edge detection pulse is cleared after finishing the current instruction, which assures that an interrupt detected between "read" and "write" stages of the Read-Modify-Write instructions will not be lost.

d) External Interrupt 3

The External Interrupt 3 can be programmed to be positive or negative edge sensitive (depending on the "i3fr" bit in "t2con" (2.4.47) SFR). When chosen edge occurs, the "iex3" flag in "ircon" (2.4.26) SFR register is set and corresponding interrupt is invoked. The low / high level must remain at least 1 clock cycle for the rising/falling edge to be detected.

Additionally the "iex3" flag can be set (and interrupt invoked) by the Compare/Capture Unit "com(0)" signal. The Compare/Capture Unit sets "iex3" flag in "ircon" (2.4.26) SFR register when compare/capture mode for CRC register is set to compare mode (flags "cocal0"=0 and "cocah0"=1 in "ccen" (2.4.7) SFR) and the value in Timer 2 becomes equal to the value of compare register "crc" (register "crc" consist of "crcl" and "crch", see 2.4.8). The external interrupt configuration (falling or rising edge sensitive) also applies here. The "i3fr" defines the edge of "com(0)" which generates interrupt, that means that it is also possible to generate interrupt when the contents of Timer 2 become not equal to the "crc" register.

The "iex3" is set by Compare/Capture Unit only when "cocal0"=0 and "cocah0"=1. in all others cases the "iex3" is set when corresponding edge on external pin occurs. It is impossible to use both External Interrupt 3 and interrupt from CCU at this same time.

The “iex3” flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on “int3” input and setting the “iex3” flag in “ircon” (2.4.26) SFR is made according to the scheme shown in Figure 123

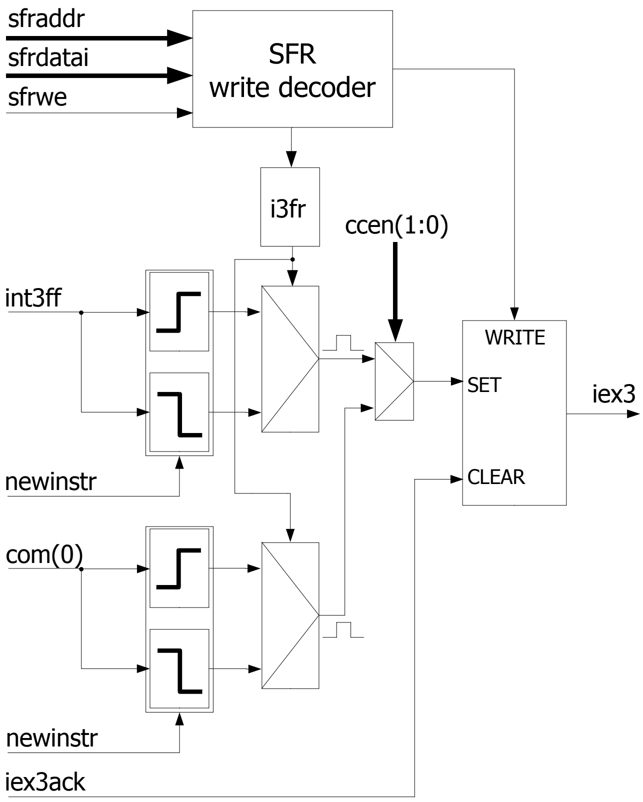


Figure 123. External Interrupt 3 Detection

e) External Interrupt 4

The External Interrupt 4 is only rising edge sensitive. When a rising edge occurs, the “iex4” flag in “ircon” (2.4.26) SFR register is set and corresponding interrupt is invoked. Low or high level at “int4” input pin must be at least 1 clock cycle long for the rising edge to be detected.

Additionally the “iex4” flag can be set (and interrupt invoked) by Compare/Capture Unit. Compare/Capture Unit sets the “iex4” flag in “ircon” SFR register when compare/capture mode for “cc1” register is set to compare mode (flags “cocal1”=0 and “cocah1”=1 in “ccen” (2.4.7) SFR) and value of Timer 2 becomes equal to the value of compare register “cc1” (register “cc1” consists of “cc1l” and “cc1h”, see 2.4.5).

The “iex4” flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on “int4” input and setting the “iex4” flag in “ircon” (2.4.26) SFR is made according to the scheme shown in Figure 124.

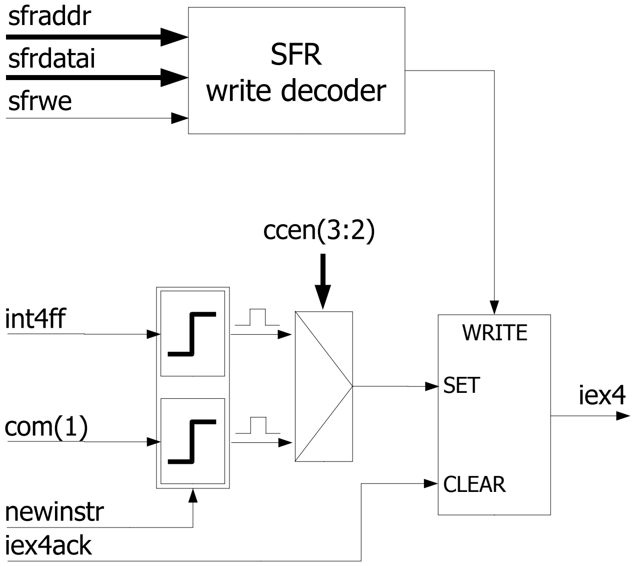


Figure 124. External Interrupt 4 Detection

f) External Interrupt 5

The External Interrupt 5 is only rising edge sensitive. When a rising edge occurs, the "iex5" flag in "ircon" (2.4.26) SFR register is set and corresponding interrupt is invoked. Low or high level at "int5" input pin must be at least 1 clock cycle long for the rising edge to be detected.

Additionally the "iex5" flag can be set (and interrupt invoked) by Compare/Capture Unit. Compare/Capture Unit sets the "iex5" flag in "ircon" SFR register when compare/capture mode for "cc2" register is set to compare mode (flags "cocal2"=0 and "cocah2"=1 in "ccen" (2.4.7) SFR) and value of Timer 2 becomes equal to the value of compare register "cc2" (register "cc2" consists of "cc2l" and "cc2h", see 2.4.5).

The "iex5" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int5" input and setting the "iex5" flag in "ircon" (2.4.26) SFR is made according to the scheme shown in Figure 125.

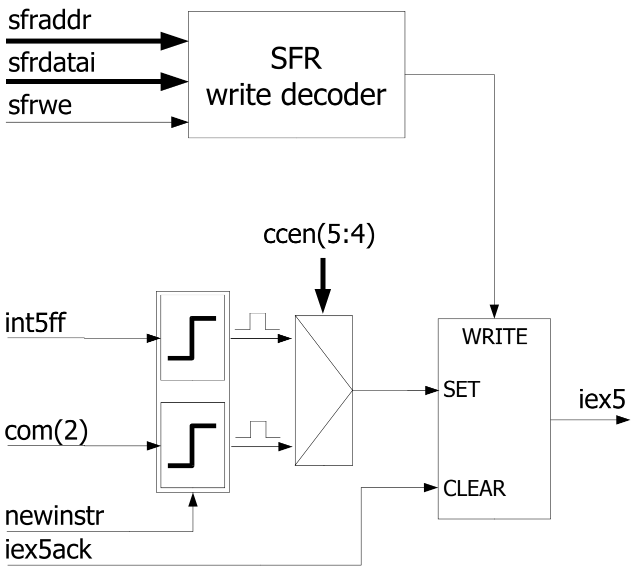


Figure 125. External Interrupt 5 Detection

g) External Interrupt 6

The External Interrupt 6 is only rising edge sensitive. When a rising edge occurs, the "iex6" flag in "ircon" (2.4.26) SFR register is set and corresponding interrupt is invoked. Low or high level at "int6" input pin must be at least 1 clock cycle long for the rising edge to be detected.

Additionally the "iex6" flag can be set (and interrupt invoked) by Compare/Capture Unit. Compare/Capture Unit sets the "iex6" flag in "ircon" SFR register when compare/capture mode for "cc3" register is set to compare mode (flags "cocal3"=0 and "cocah3"=1 in "ccen" (2.4.7) SFR) and value of Timer 2 becomes equal to the value of compare register "cc3" (register "cc3" consists of "cc3l" and "cc3h", see 2.4.5).

The "iex6" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int6" input and setting the "iex6" flag in "ircon" (2.4.26) SFR is made according to the scheme shown in Figure 126.

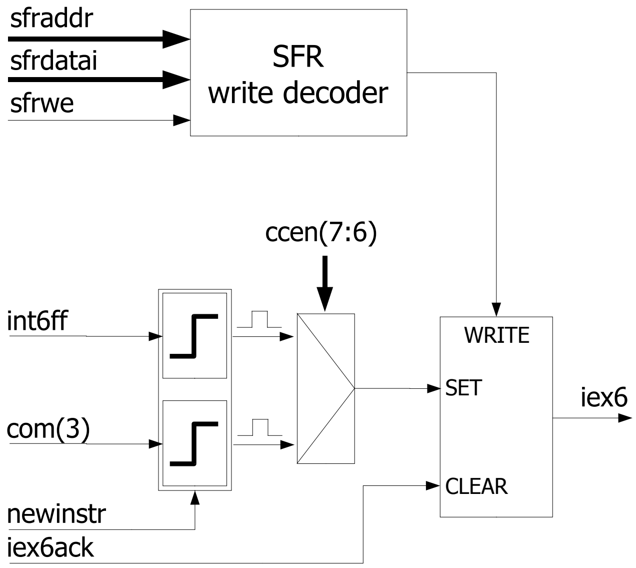


Figure 126. External Interrupt 6 Detection

h) External Interrupt 7

The External Interrupt 7 is only rising edge sensitive. When a rising edge occurs, the "iex7" flag in "ircon" (2.4.26) SFR register is set and corresponding interrupt is invoked. Low or high level at "int7" input pin must be at least 1 clock cycle long for the rising edge to be detected.

The "iex7" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int7" input and setting the "iex7" flag in "ircon" (2.4.26) SFR is made according to the scheme shown in Figure 127.

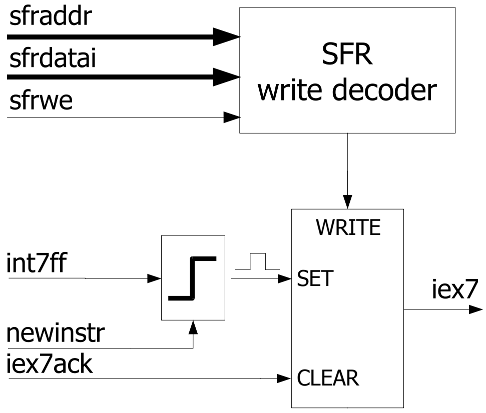


Figure 127. External Interrupt 7 Detection

i) External Interrupt 8

The External Interrupt 8 is only rising edge sensitive. When a rising edge occurs, the "iex8" flag in "ircon2" (2.4.27) SFR register is set and corresponding interrupt is invoked. Low or high level at "int8" input pin must be at least 1 clock cycle long for the rising edge to be detected.

The "iex8" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int8" input and setting the "iex8" flag in "ircon2" (2.4.27) SFR is made according to the scheme shown in Figure 128.

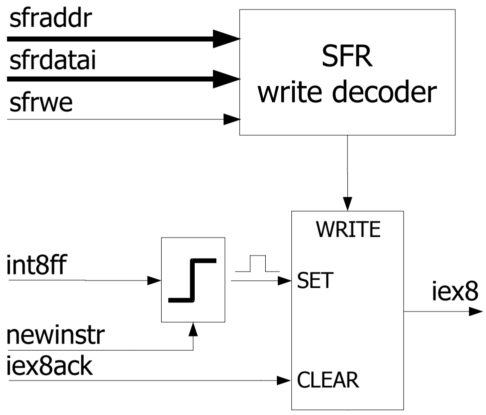


Figure 128. External Interrupt 8 Detection

j) External Interrupt 9

The External Interrupt 9 is only rising edge sensitive. When a rising edge occurs, the "iex9" flag in "ircon2" (2.4.27) SFR register is set and corresponding interrupt is invoked. Low or high level at "int9" input pin must be at least 1 clock cycle long for the rising edge to be detected.

The "iex9" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int9" input and setting the "iex9" flag in "ircon2" (2.4.27) SFR is made according to the scheme shown in Figure 129.

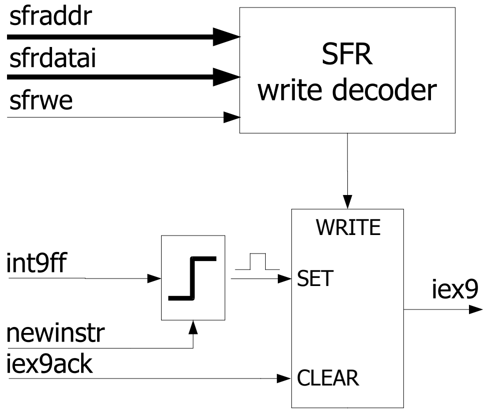


Figure 129. External Interrupt 9 Detection

k) External Interrupt 10

The External Interrupt 10 is only rising edge sensitive. When a rising edge occurs, the "iex10" flag in "ircon2" (2.4.27) SFR register is set and corresponding interrupt is invoked. Low or high level at "int10" input pin must be at least 1 clock cycle long for the rising edge to be detected.

The "iex10" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int10" input and setting the "iex10" flag in "ircon2" (2.4.27) SFR is made according to the scheme shown in Figure 130.

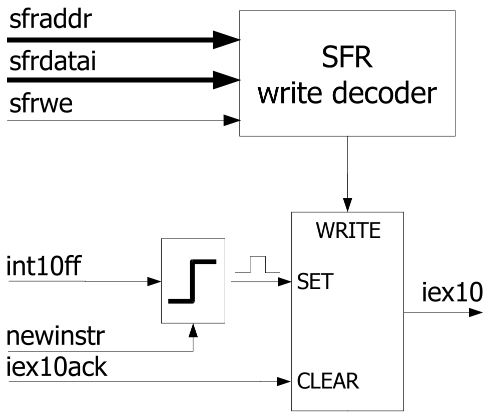


Figure 130. External Interrupt 10 Detection

l) External Interrupt 11

The External Interrupt 11 is only rising edge sensitive. When a rising edge occurs, the "iex11" flag in "ircon2" (2.4.27) SFR register is set and corresponding interrupt is invoked. Low or high level at "int11" input pin must be at least 1 clock cycle long for the rising edge to be detected.

The "iex11" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int11" input and setting the "iex11" flag in "ircon2" (2.4.27) SFR is made according to the scheme shown in Figure 131.

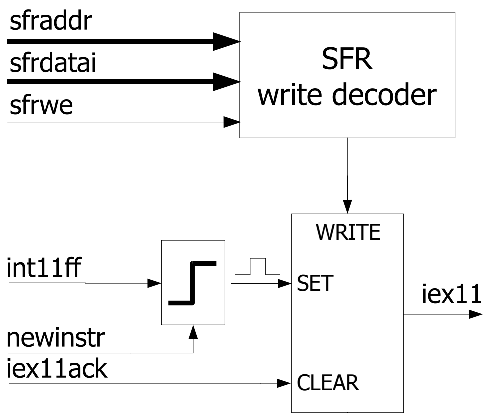


Figure 131. External Interrupt 11 Detection

m) External Interrupt 12

The External Interrupt 12 is only rising edge sensitive. When a rising edge occurs, the "iex12" flag in "ircon2" (2.4.27) SFR register is set and corresponding interrupt is invoked. Low or high level at "int12" input pin must be at least 1 clock cycle long for the rising edge to be detected.

The "iex12" flag is cleared by hardware automatically when invoked service routine is vectored to.

The hardware edge detection on "int12" input and setting the "iex12" flag in "ircon2" (2.4.27) SFR is made according to the scheme shown in .

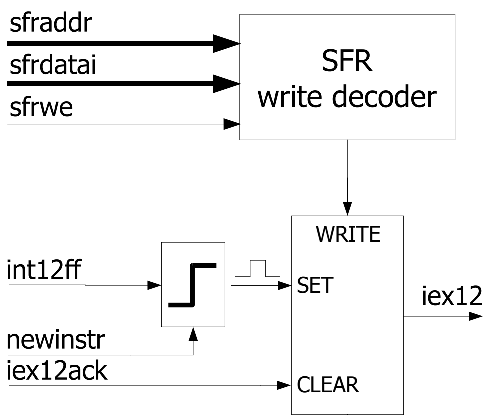


Figure 132. External Interrupt 12 Detection

5.17. I2C

5.17.1 Overview

The **I2C** subcomponent is the I2C Bus Controller which provides an interface that meets the Philips I2C bus specification and supports all transfer modes from and to the I2C bus.

The I2C bus uses two wires to transfer information between devices connected to the bus: "scl" (serial clock line) and "sda" (serial data line).

The I2C logic handles bytes transfer autonomously. It also keeps track of serial transfers, and a status register ("i2csta", 2.4.17) reflects the status of the I2C Bus Controller and the I2C bus.

5.17.2 Pin Description

Table 103. I2C Pin Description

Name	Type	Polarity Bus size	Description
clk	I	Rise	**Clock** Clock input for all internal synchronous logic
rst	I	High	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
bclk	I	Rise	**Baud rate clock** Pulse for transmission speed control, internally synchronized with the "clk" input
si	O	High	**Serial transmission interrupt** Active high when service is requested by the I2C component
I2C interface			
scli	I	Level	I2C clock input
sdai	I	Level	I2C data input
sclo	O	Level	I2C clock output
sdao	O	Level	I2C data output
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs ("i2cdat", "i2ccon", "i2cadr")
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"
i2cdat_o	O	8	**I2C Data Register output**
i2cadr_o	O	8	**I2C Own Slave Address Register output**
i2ccon_o	O	8	**I2C Control Register output**
i2csta_o	O	8	**I2C Status Register output**

5.17.3 Block Diagram

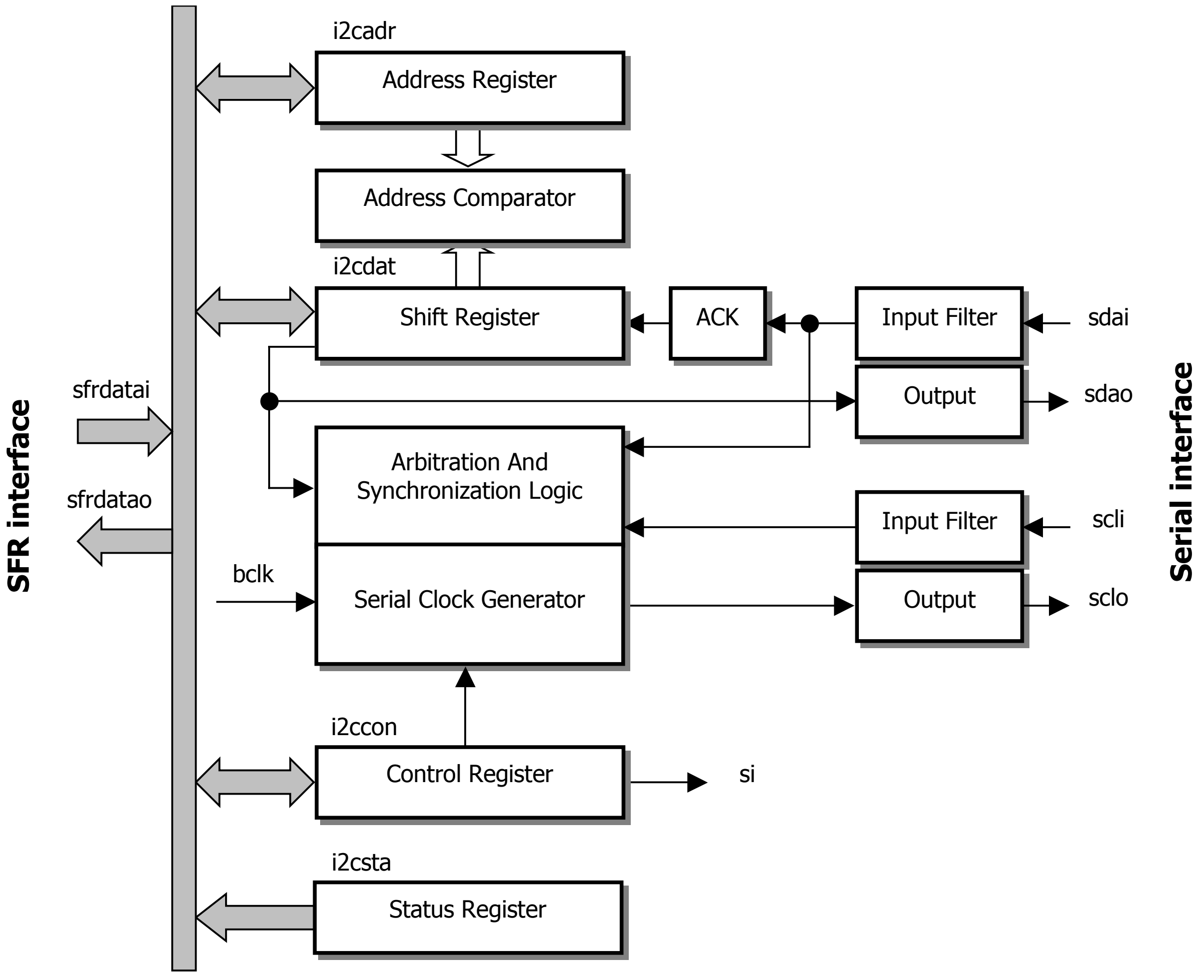


Figure 133. I2C Block Diagram

5.17.4 Description

The I2C bus uses two wires to transfer information between devices connected to the bus: "scl" (serial clock line) and "sda" (serial data line). The I2C component requires the use of external open-drain buffers since it has only unidirectional ports. The "sda" and "scl" lines refered further are the actual I2C bus signals, while the I2C component is connected to them with the use of open-drains ("sdao" as output and "sdai" as input, "sclo" as output and "scli" as input). Each device connected to the bus is software addressable by a unique address. The I2C is a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer. The filtering logic rejects spikes on the bus data line to preserve data integrity.

a) Operating Modes

The I2C component performs 8-bit oriented, bi-directional data transfers up to 100 kbit/s in the standard mode or up to 400 kbit/s in the fast mode and may operate in the following four modes:

b) Master Transmitter Mode:

Serial data output through "sdao" while "sclo" outputs the serial clock.

c) Master Receiver Mode:

Serial data is received via "sdai" while "sclo" outputs the serial clock.

d) Slave Receiver Mode

Serial data and the serial clock are received through "sdai" and "scli".

e) Slave Transmitter Mode

Serial data is transmitted via "sdao" while the serial clock is input through "scli".

f) Arbitration and Synchronization Logic

In the master mode, the arbitration logic checks that every transmitted high state ('1') on "sdao" actually appears as high state ('1') on the I2C bus "sda". If another device on the bus overrides high and pulls the "sda" line low ('0'), arbitration is lost and the I2C immediately changes from master transmitter to slave receiver.

The synchronization logic synchronizes the serial clock generator with the clock pulses on the "scli" line from another device.

g) Serial Clock Generator

This programmable clock pulse generator provides the "sclo" clock pulses when the I2C is in the master mode. The clock generator is suppressed when the I2C is in the slave mode.

The function of the clock generator is controlled by bits "cr0", "cr1" and "cr2" of "i2ccon" register (2.4.15). The table below shows the possible rates of "clko" in the master mode.

The "bclk" input referenced in the table is connected to the Timer 1 overflow output. That means the baud rate of the I2C can be controlled by the Timer 1.

Table 104. I2C Clock Rate Bit Settings

cr2	cr1	cr0	Bit frequency				Clk divided by
			6 MHz	12 MHz	16 MHz	24 MHz	
0	0	0	23	47	63	92	256
0	0	1	27	54	71	108	224
0	1	0	31	63	83	124	192
0	1	1	37	75	100	148	160
1	0	0	6.25	12.5	17	25	960
1	0	1	50	100	133	200	120
1	1	0	100	200	266	400	60
1	1	1	"bclk" input divided by 8				

h) Input Filter

Input signals are synchronized with clock ("clk"), and spikes shorter than three clock periods are filtered out. Each filter consists of three flip-flops. The first one is used to latch the input directly, while the other two form a shift register which is loaded from the first one. When the state of the 2nd and 3rd flip-flop is either "11" or "00", an internal filtered signal is set or reset, respectively.

i) Address Comparator

The received 7-bit slave address is compared with the I2C component own slave address. The own slave address can be programmed using the "i2caddr" register (see 2.4.14). Also the first received byte is compared with the general call address (00H), depending on the "gc" bit of "i2caddr" register (2.4.14). If equality is found, the "si" bit of the "i2ccon" register (2.4.15) is set and an interrupt is requested.

j) Interrupt Generation

The "si" flag of the "i2ccon" register (2.4.15) is set by hardware when one of 25 out of 26 possible I2C states is entered (2.4.17). The only state that does not set the "si" is state F8h, which indicates that no relevant state information is available. The "si" flag must be cleared by software. in order to clear the "si" bit, '0' must be written to this bit. Writing a '1' to si bit does not change value of the "si".

The I2C interrupt vector is shared with the External Interrupt 7. The "si" signal is OR-ed with the External Interrupt 7 edge flag, before it comes into the Interrupt Controller (ISR). to determine the actual source of that interrupt, the "si" flag has to be investigated by the interrupt service routine. Since the External Interrupt 7 flag is automatically cleared after vectoring to the service subroutine, only the state of the "si" flag brings the information of the actual source of the interrupt.

k) Special Function Registers

The microprocessor interfaces to the I2C component via the following four special function registers: "i2ccon" (control register, 2.4.15), "i2csta" (status register, 2.4.17), "i2cdat" (data register, 2.4.16) and "i2cadr" (own slave address register, 2.4.14).

The "i2cadr" register contains the Own Slave Address of the I2C component, and the "gc" flag which enables the recognition of a general call address.

The "i2ccon" register contains the global I2C enable bit "ens1", clock rate setting bits ("cr0", "cr1", "cr2", see Table 11. I2C2CON Register). It also provides flags to initiate sending START or STOP conditions to the I2C bus ("sta", "sto" bits), a flag controlling the ACK bit in I2C transmission after receiving own slave address or general call address or after receiving data either in master or slave mode ("aa" – assert acknowledge flag). Finally the "i2ccon" provides the interrupt request flag "si" which is set by hardware when a change of the main controlling FSM is detected. See the "i2csta" register description for FSM details.

The "i2cdat" register contains a byte to be transmitted through I2C bus or a byte which has just been received through I2C bus. The "i2cdat" register is not shadowed or double buffered so the MCU should only read it when an I2C interrupt occurs.

The "i2csta" register reflects the state of the main FSM of the I2C component. The three least significant bits of this register are always zero. There are 26 possible status codes, which are presented in Table 105 … Table 109. When one of the 25 out of 26 possible I2C FSM states is entered, an interrupt is requested. The only state that does not generate an interrupt is the F8h state.

In the table below, referring to "SLA" means slave address, "R" means R/W bit=1 tramsferred together with the slave address, "W" means R/W bit=0 transferred together with the slave address.

Table 105. I2C Status in Master Transmitter Mode

Status code	Status of the I2C	Application software response: to/from I2CDAT	to I2CCON: sta	sto	si	aa	Next action taken by the I2C hardware
08H	START condition has been transmitted	Load SLA+W	X	0	0	X	SLA+W will be transmitted ACK will be received
10H	Repeated START condition has been transmitted	Load SLA+W	X	0	0	X	As above
		or Load SLA+R	X	0	0	X	SLA+R will be transmitted I2C will be switched to "master receiver" mode
18H	SLA+W has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK will be received
		or no action	1	0	0	X	Repeated START will be transmitted;
		or no action	0	1	0	X	STOP condition will be transmitted; the "sto" flag will be reset
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; the "sto" flag will be reset
20H	SLA+W has been transmitted; "not ACK" has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK will be received
		or no action	1	0	0	X	Repeated START will be transmitted
		or no action	0	1	0	X	STOP condition will be transmitted; the "sto" flag will be reset
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; the "sto" flag will be reset
28H	Data byte in i2cdat has been transmitted; ACK has been received	Load data byte	0	0	0	X	Data byte will be transmitted; ACK bit will be received
		or no action	1	0	0	X	Repeated START will be transmitted
		or no action	0	1	0	X	STOP condition will be transmitted; the "sto" flag will be reset
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; sto flag will be reset

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to I2CCON				
			sta	sto	si	aa	
30H	Data byte in i2cdat has been transmitted	Data byte	0	0	0	X	Data byte will be transmitted; ACK will be received
		or no action	1	0	0	X	Repeated START will be transmitted;
		or no action	0	1	0	X	STOP condition will be transmitted; sto flag will be reset
		or no action	1	1	0	X	STOP condition followed by a START condition will be transmitted; sto flag will be reset
38H	Arbitration lost in SLA+R/W or data bytes	No action	0	0	0	X	I2C bus will be released; the "not addressed slave" state will be entered
		or no action	1	0	0	X	A START condition will be transmitted when the bus becomes free

Table 106. I2C Status in Master Receiver Mode

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
08H	START condition has been transmitted	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK will be received
10H	Repeated START condition has been transmitted	Load SLA+R	X	0	0	X	As above
		or Load SLA+W	X	0	0	X	SLA+W will be transmitted; I2C will be switched to "master transmitter" mode
38H	Arbitration lost in "not ACK" bit	No action	0	0	0	X	I2C bus will be released; I2C will enter a "slave" mode
		or no action	1	0	0	X	A start condition will be transmitted when the bus becomes free
40H	SLA+R has been transmitted; ACK has been received	No action	0	0	0	0	Data byte will be received; not ACK will be returned
		or no action	0	0	0	1	Data byte will be received; ACK will be returned

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
48H	SLA+R has been transmitted; "not ACK" has been received	No action	1	0	0	X	Repeated START condition will be transmitted
		or no action	0	1	0	X	STOP condition will be transmitted; the "sto" flag will be reset
		or no action	1	1	0	X	STOP condition followed by START condition will be transmitted; the "sto" flag will be reset
50H	Data byte has been received; ACK has been returned	Read data byte or	0	0	0	0	Data byte will be received; "not ACK" will be returned
		read data byte	0	0	0	1	Data byte will be received; ACK will be returned
58H	Data byte has been received; "not ACK" has been returned	Read data byte or	1	0	0	X	Repeated START condition will be transmitted
		read data byte or	0	1	0	X	STOP condition will be transmitted; the "sto" flag will be reset
		read data byte	1	1	0	X	STOP condition followed by START condition will be transmitted; the "sto" flag will be reset

Table 107. I2C Status in Slave Receiver Mode

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
60H	Own SLA+W has been received; ACK has been returned	No action	X	0	0	0	Data byte will be received and "not ACK" will be returned
		or no action	X	0	0	1	Data byte will be received and ACK will be returned
68H	Arbitration lost in SLA+R/W as master; own SLA+W has been received, ACK returned	No action	X	0	0	0	Data byte will be received and "not ACK" will be returned
		or no action	X	0	0	1	Data byte will be received and ACK will be returned
70H	General call address (00H) has been received; ACK has been returned	No action	X	0	0	0	Data byte will be received and "not ACK" will be returned
		or no action	X	0	0	1	Data byte will be received and ACK will be returned

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
78H	Arbitration lost in SLA+R/W as master; general call address has been received, ACK returned	No action	X	0	0	0	Data byte will be received and "not ACK" will be returned
		or no action	X	0	0	1	Data byte will be received and ACK will be returned
80H	Previously addressed with own SLV address; DATA has been received; ACK returned	Read data byte	X	0	0	0	Data byte will be received and "not ACK" will be returned
		or read data byte	X	0	0	1	Data byte will be received and ACK will be returned
88H	Previously addressed with own SLA; DATA byte has been received; "not ACK" returned	Read data byte	0	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address
		or read data byte	0	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized
		or read data byte	1	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address; START condition will be transmitted when the bus becomes free
		or read data byte	1	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized; START condition will be transmitted when the bus becomes free
90H	Previously addressed with general call address; DATA has been received; ACK returned	Read data byte	X	0	0	0	Data byte will be received and "not ACK" will be returned
		or read data byte	X	0	0	1	Data byte will be received and ACK will be returned

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
98H	Previously addressed with general call address; DATA has been received; ACK returned	Read data byte	0	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address
		or read data byte	0	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized
		or read data byte	1	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address; START condition will be transmitted when the bus becomes free
		or read data byte	1	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized; START condition will be transmitted when the bus becomes free
A0H	STOP condition or repeated START condition has been received while still addressed as SLV/REC or SLV/TRX	No action	0	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address
		or no action	0	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized
		or no action	1	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address; START condition will be transmitted when the bus becomes free
		or no action	1	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized; START condition will be transmitted when the bus becomes free

Table 108. I2C Status in Slave Transmitter Mode

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
A8H	Own SLA+R has been received; ACK has been returned	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK will be received
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received
B0H	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK will be received
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received
B8H	Data byte has been transmitted; ACK has been received	Load data byte	X	0	0	0	Last data byte will be transmitted and ACK will be received
		or load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received
C0H	Data byte has been transmitted; not ACK has been received	No action	0	0	0	0	Switched to “not addressed slave” mode; no recognition of own slave address or general call address
		or no action	0	0	0	1	Switched to “not addressed slave” mode; own slave address or general call address will be recognized
		or no action	1	0	0	0	Switched to “not addressed slave” mode; no recognition of own slave address or general call address; START condition will be transmitted when the bus becomes free
		or no action	1	0	0	1	Switched to “not addressed slave” mode; own slave address or general call address will be recognized; START condition will be transmitted when the bus becomes free

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
C8H	Last data byte has been transmitted; ACK has been received	No action	0	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address
		or no action	0	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized
		or no action	1	0	0	0	Switched to "not addressed slave" mode; no recognition of own slave address or general call address; START condition will be transmitted when the bus becomes free
		or no action	1	0	0	1	Switched to "not addressed slave" mode; own slave address or general call address will be recognized; START condition will be transmitted when the bus becomes free

Table 109. I2C Status - Miscellaneous States

Status code	Status of the I2C	Application software response					Next action taken by the I2C hardware
		to/from I2CDAT	to i2CCON				
			sta	sto	si	aa	
F8H	No relevant state information available; si=0	No action	No action				Wait or proceed current transfer
00H	Bus error during MST or selected slave modes	No action	0	1	0	X	Only the internal hardware is affected in the "master" or "addressed slave" modes. in all cases, the bus is released and I2C is switched to the "not addressed slave" mode. The "sto" flag is reset.

5.18. SEC_I2C

5.18.1 Overview

The Secondary I2C subcomponent is the same I2C Bus Controller as described in section 5.17, with only the SFR locations different.

It can only be implemented when the basic I2C interface is also implemented.

5.18.2 Pin Description

Table 110. Secondary I2C Pin Description

Name	Type	Polarity Bus size	Description
clk	I	Rise	**Clock** Clock input for all internal synchronous logic
rst	I	High	**Hardware reset input** The subcomponent is reset when this pin is held high for at least one clock cycle
bclk2	I	Rise	**Baud rate clock** Pulse for transmission speed control, internally synchronized with the "clk" input
si2	O	High	**Serial transmission interrupt** Active high when service is requested by the I2C component
Secondary I2C interface			
scl2i	I	Level	I2C clock input
sda2i	I	Level	I2C data input
scl2o	O	Level	I2C clock output
sda2o	O	Level	I2C data output
Special Function Register interface			
sfrdatai	I	8	**SFR data bus input** Data to be written to internal SFRs ("i2c2dat", "i2c2con", "i2c2adr")
sfraddr	I	7	**SFR Address bus** Contains the address of SFR being read or written
sfrwe	I	High	**SFR Write enable** Enables write to the SFR pointed by "sfraddr"
i2c2dat_o	O	8	**Secondary I2C Data Register output**
i2c2adr_o	O	8	**Secondary I2C Own Slave Address Register output**
i2c2con_o	O	8	**Secondary I2C Control Register output**
i2c2sta_o	O	8	**Secondary I2C Status Register output**

5.18.3 Block Diagram

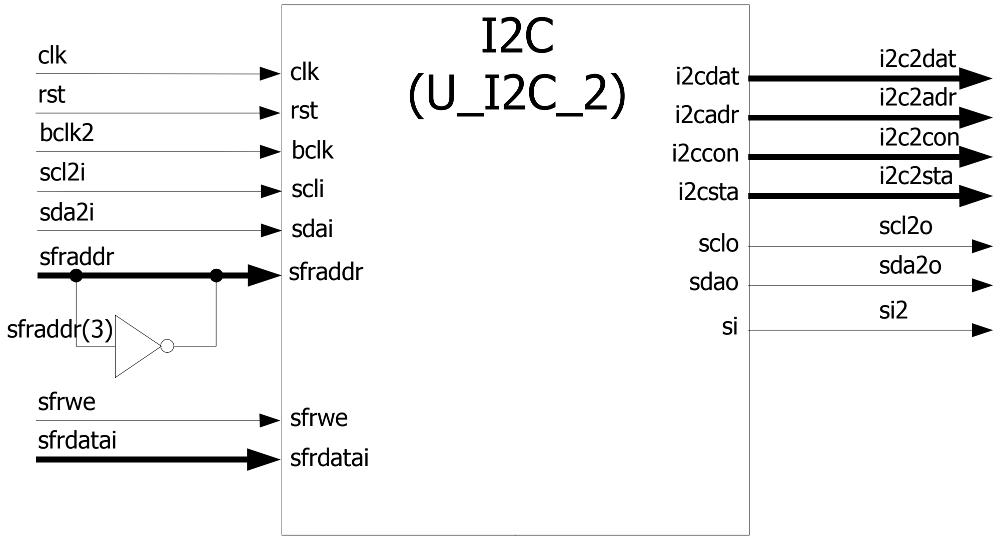


Figure 134. Secondary I2C Block Diagram

5.18.4 Description

The Secondary I2C unit instantiates the I2C module described in section 5.17.

It works identical with the only exception that the Special Function Registers "i2c2dat" (2.4.16), "i2c2adr" (2.4.14), "i2c2con" (2.4.15) and "i2c2sta" (2.4.17) are located respectively at addresses: 0xD2, 0xD3, 0xD4 and 0xD5.

The interrupt generation of the Secondary I2C is identical as in the I2C module. The interrupt request output is OR-ed with interrupt requests from I2C and External Interrupt 7. to recognize the source of interrupt, when vectored to address 0x0043 the "i2csta" and "i2c2sta" registers must be checked.

5.19. SPI_MS

5.19.1 Overview

The SPI_MS module allows full-duplex, synchronous, serial communication between the MCU and peripherals, including other MCUs. It is obvious that the MCU and any peripherals must include SPI module.

The module may be programmed to work as master or as slave device.

The SPI_MS provides the following features:

- Full duplex mode
- Three wire synchronous transfers
- Master or Slave mode
- Seven SPI Master baud rates
- Slave Clock rate up to Fclk/4
- Serial clock with programmable polarity and phase
- Master Mode fault error flag with MCU interrupt capability

- Write collision flag protection
- 8-bit data transmitted Most Significant Bit (MSB) first, Least Significant Bit (LSB) last
- 8-bit Slave Select Output port to control external slave devices
- Special Function Registers interface to the host CPU
- No bi-directional ports; standard SPI pins to be externally connected to 3-state buffers

5.19.2 Pin Description

Table 111. SPI_MS Pin Description

Name	Type	Polarity Bus size	Description
clk	I	Rise	**Clock** Pulse for all synchronous circuits
rst	I	High	**Reset** Puts all synchronous logic into a known state.
sfraddr	I	7	**Special Function Registers address bus** Selects which internal SFR is currently accessed.
sfrdatai	I	8	**Special Function Registers input data bus** Data to be written to internal SFRs
sfrwe	I	High	**Special Function Registers write enable** Enables write of "sfrdatai" to internal SFR selected by "sfraddr"
sfroe	I	High	**Special Function Registers read enable** Indicates that internal SFR selected by "sfraddr" is read.
spen	I	High	**Serial Peripheral Enable** Enables the SPI. When 1, the component is enabled and transmission may be initiated. When 0, resets the internal FSM and forces "scko", scktri", "misoo", misotri", "mosio" and "mositri" outputs to 0. It is connected to "spcon.6" bit at the top level of the core.
mstr	I	High	**Serial Peripheral Master** Configures the SPI as a Master or a Slave. When 1, the component is configured as Master otherwise it is configured as Slave. It is connected to "spcon.4" bit at the top level of the core.
cpol	I	-	**Clock Polarity** Configures the idle state of "scko" serial clock when SPI is enabled (when disabled, "scko" is in high level). When 1, the "scko" output is set (i.e. "scko" = '1') otherwise it is cleared (i.e. "scko" = '0'). It is connected to "spcon.3" bit at the top level of the core.
cpha	I	-	**Clock Phase** Configures the data sampling point. When 0, data is sampled when "sck" goes to active state (see "cpol"). When 1, data is sampled when "sck" goes to idle state (see "cpol"). It is connected to "spcon.2" bit at the top level of the core.
scki	I	Rise/Fall	**Serial Clock Input** Pulse for shifting input data when in Slave Mode. The maximum rate allowed is the "clk" frequency / 4.
ssn	I	Low	**Slave Select Input** Activates the SPI module as a Slave. When SPI was configured as a Master, a Mode Fault error is generated.
misoi	I	-	**Master Input** Data to be shifted into a receive register in Master Mode.

Name	Type	Polarity Bus size	Description
mosii	I	-	**Slave Input** Data to be shifted into a receive register in Slave Mode.
spcon	O	8	**Serial Peripheral Control Register output** Direct output of "spcon" register.
spsta	O	8	**Serial Peripheral Status Register output** Direct output of "spsta" register.
spdat	O	8	**Serial Peripheral Data Register output** Direct output of "spdat" receive data buffer register.
sfrdatao	O	8	**Special Function Registers bus output** Outputs one of "spcon", "spsta", "spdat" registers depending on "sfraddr". Note that, unlike "spcon" output, the "spcon" register is here combined of "cpha", "cpol", "mstr" and "spen" inputs.
scko scktri	O O	Rise/Fall High	**Serial Clock Output** **Serial Clock Output Tri-state** Pulse for external slaves for shifting data, generated in Master Mode. The "tri" output enables the on-chip, off-core tri-state output buffer. Together with "scki" those ports form a bi-directional "sck" pad.
misoo misotri	O O	- High	**Slave Output** **Slave Output Tri-state** Data output when the SPI is configured in Slave Mode. The "tri" output enables the on-chip, off-core tri-state output buffer. Together with "misoi" those ports form a bi-directional "miso" pad.
mosio mositri	O O	- High	**Master Output** **Master Output Tri-state** Data output when the SPI is configured in Master Mode. The "tri" output enables the on-chip, off-core tri-state output buffer. Together with "mosii" those ports form a bi-directional "mosi" pad.
intspi	O	High	**SPI Interrupt Request** Request for the MCU to generate an interrupt.
spssn	O	8	**Slave select output register**

5.19.3 Block Diagram

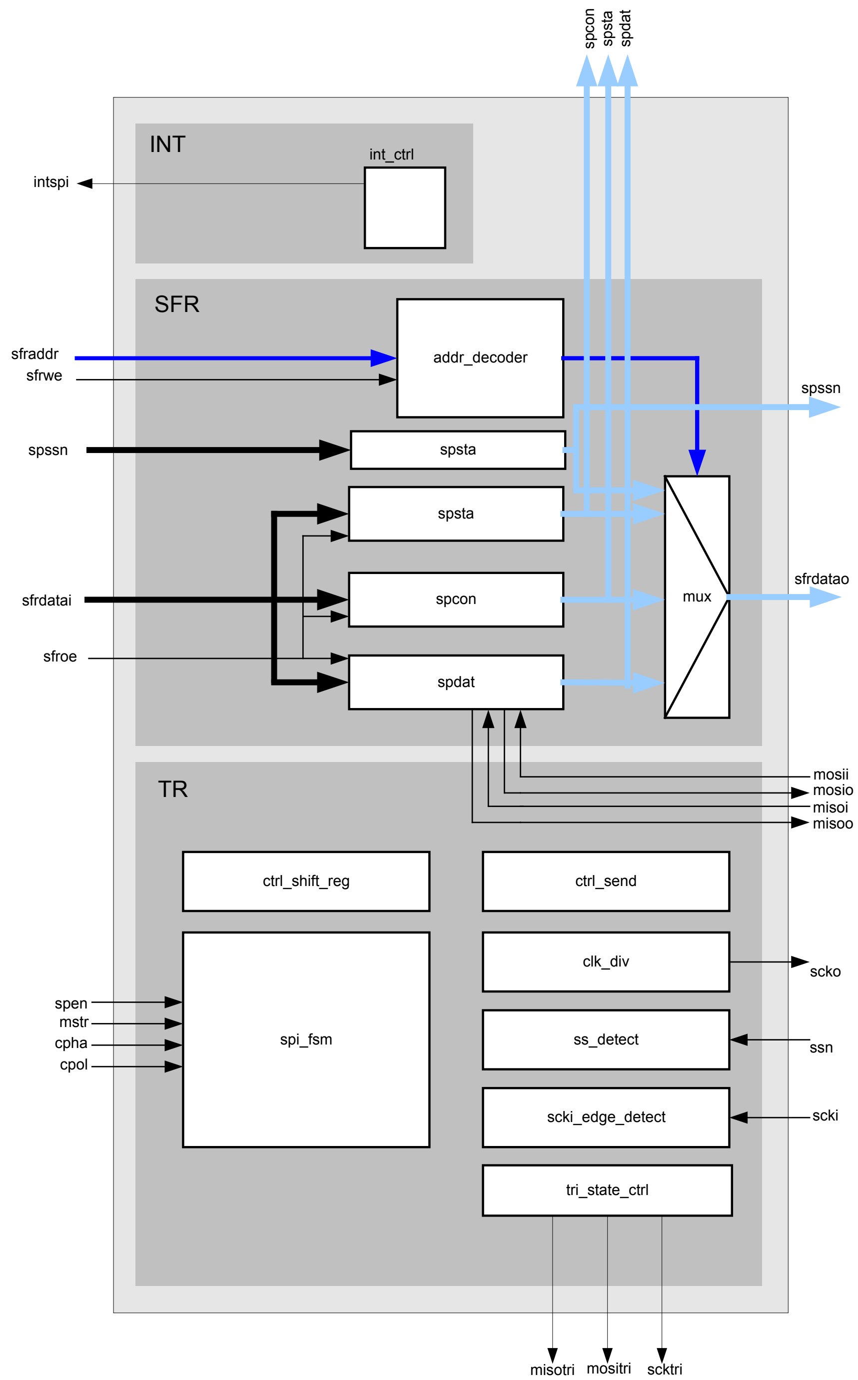


Figure 135. SPI_MS Block Diagram

5.19.4 Description

The component communicates with host microprocessor through SFR interface (i.e. "sfrdatai", "sfraddr", "sfrwe", "sfroe") and INT interface (i.e. "intspi"). Communication with other devices, which include the SPI module, is realized through TR interface (i.e. "mosi" group, "miso" group, "sck" group).

Functional blocks of SPI_MS module:

- INT – interrupt control block

- SFR – Special Functional Register block
- TR – Transmit block

- The SFR block controls the write/read operations on SFR registers of SPI_MS module. It contains the following:
  - address decoder,
  - Special Function Registers – SPCON, SPSTA, SPDAT,
  - output multiplexor.
- The TR block controls the SPI transmission process. It is composed of the following:
  - the Finite State Machine which plays a key role in operation of the SPI_MS module; it controls the Master or Slave functionality
  - system clock counter/divider, which is used to generate the SPI Master clock "scko"; the Master clock is selected from one of seven clock rates i.e. The system clock divided by 2, 4, 8, 16, 32, 64 or 128
  - rising and falling edge detector on "scki" input pin; it is used only in Slave mode
  - transmission end detector
  - level and falling edge detector on "ssn" input pin
  - data shift register.
- The INT block generates interrupt request upon "spif" and "modf" flags.

a) Special Function Registers

There are three special function registers in the SPI_MS.

Table 112. SPI_MS Special Function Registers

Register	Location	Reset value	Description
SPCON	E2h	14H	Serial Peripheral Control register
SPSTA	E1h	00H	Serial Peripheral Status register
SPDAT	E3h	00H	Serial Peripheral Data register
SPSSN	E4h	FFH	Serial Peripheral Slave Select register

The "spcon" (2.4.42) register is used in configuration of the SPI_MS module. It controls the master clock output rate (bits "spr0", "spr1" and "spr2"), the clock polarity ("cpol") and phase ("cpha"), configures the SPI_MS either as master or slave ("mstr" bit), enables or disables the "ssn" input ("ssdis" bit) and enables or disables the whole SPI_MS component ("spen" bit).

The "spsta" (2.4.45) register reflects the current status of the SPI_MS module. The "spif" flag informs that there is a transfer in progress or a transfer has finished. The "wcol" flag indicates that a write collision on the "spdat" has occured, i.e. The "spdat" register was written through the SFR interface while there was a transfer on the SPI interface. The "serr" bit informs that the "ssn" input was removed before the end of receive sequence when the SPI_MS was configured as slave. Finally, the "modf" bit notifies when the state of the "ssn" input is in conflict with the actual mode settings (i.e. when "ssn"=0 and the SPI_MS is configured as a master).

The "spdat" (2.4.43) register is used during transmission process. Data from this register can be sent through the TR (Transmit/Receive) interface, i.e. byte of data begins shifting out on

output pin ("mosio" - in Master mode, "misoo" - in Slave mode). Simultaneously, another byte shifts in from input pin ("misoi" - in Master mode, "mosii" - in Slave mode). After transmission process is completed the received data can be read from "spdat" register.

The "spssn" (2.4.44) register is used to control up to 8 external slave devices connected to the core. It can be also used as a general purpose 8-bit output port.

b) Tr Interface

Tha main SPI interface is controlled by the TR (Transmit/Receive) block (see Figure 135). Figure 136 shows the general format of data transmission. Depending on the settings of SPI_MS module, the bits of data are sent in turn on rising edge ("cpol" = '0') or on falling edge ("cpol"= '1') of master clock ("scko"). Data are received at the falling edge ("cpol"='0') or rising edge ("cpol"='1') of master clock ("scko"). This applies either for master or slave transmitter/receiver, assuming that "scko" is the main clock of the transmission. If "cpha" bit is set, the first bit (MSB) will be sent on the "mosio"/"misoo" at the first active edge of "scko". If "cpha" bit is cleared, the first bit (MSB) will be sent half a period of "scko" signal before active edge of this signal. in addition, the data input ("misoi" for master, "mosii" for slave) is sampled in the half of each bit transmitted, at the opposite edge of the clock at which data are shifted out to "mosio" output.

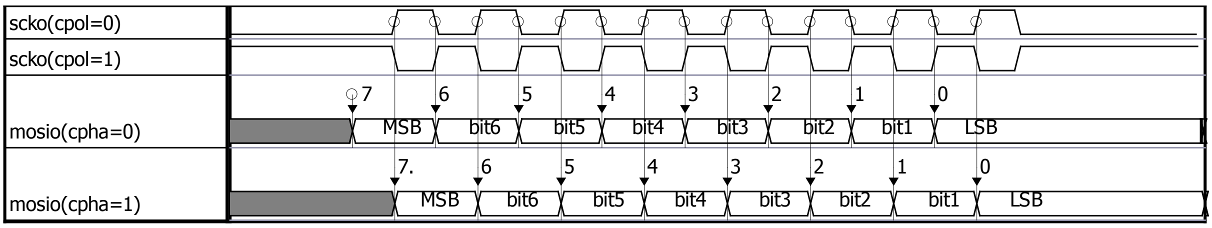


Figure 136. SPI_MS Transmitter Frame Format

The TR interface consists of four inputs and six outputs:

- Inputs
  - scki – serial clock
  - ssn – slave select
  - misoi – master input slave output
  - mosii – master output slave input
- Outputs:
  - scko – serial clock
  - scktri – tri-state buffer enable of sck pin
  - misoo – master input slave output
  - misotri – tri-state buffer enable of miso pin
  - mosio – master output slave input
  - mositri – tri-state buffer enable of mosi pin

These inputs and outputs are divided in three groups:

- "miso" group

The "misoo", "misotri" and "misoi" are dedicated to an off-core connection with a tri-state buffer to provide an external bi-directional port "miso".

- "mosi" group

The “mosio”, “mositri” and “mosii” are dedicated to an off-core connection with a tri-state buffer to provide an external bi-directional port “mosi”.

- “sck” group

The “scko”, “scktri” and “scki” are dedicated to an off-core connection with a tri-state buffer to provide an external bi-directional port “sck”.

-
- MASTER MODE

In master mode (the “mstr” bit of “spcon” register is set) the SPI_MS waits on write operation to “spdat” register. If write operation to “spdat” register is done, transmission is started. Data shifts out on “mosio” pin at the “scko” serial clock transition (“send_edge”) . Simultaneously, another data byte shifts in from the slave on master's “misoi” pin (“capture_edge”).

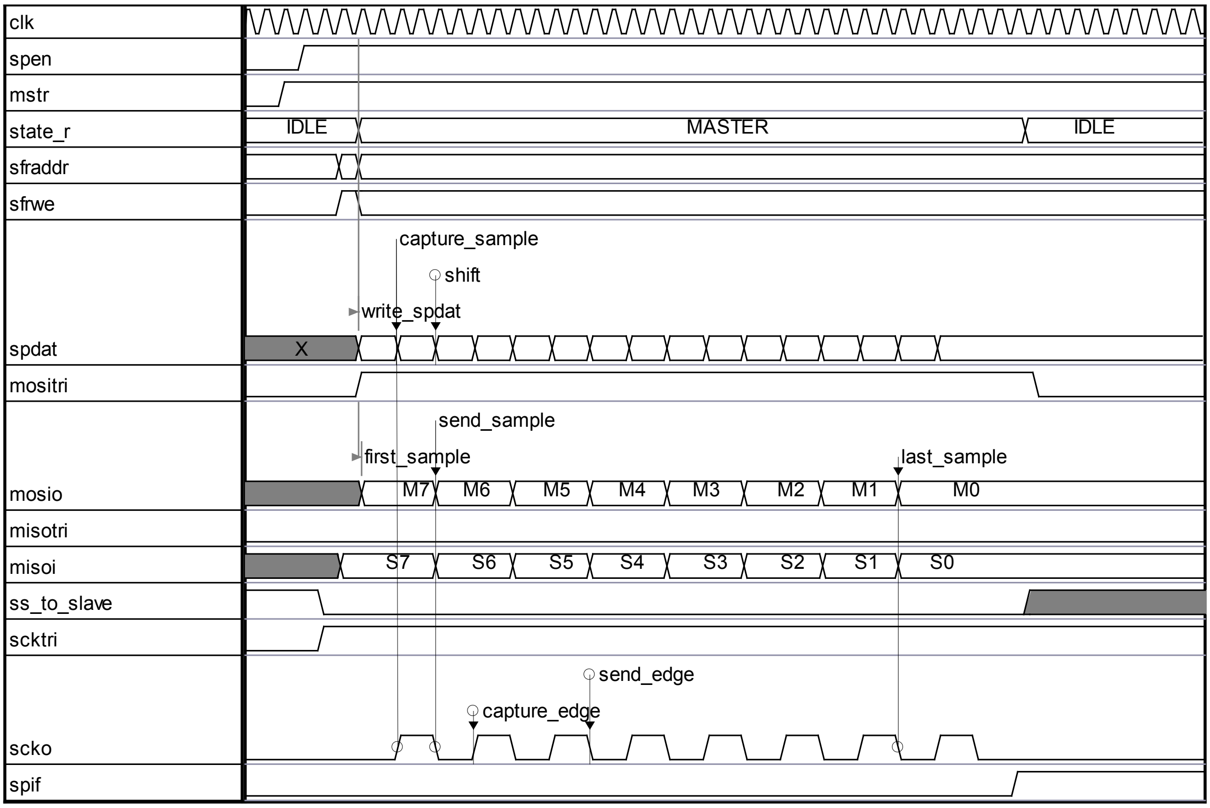


Figure 137. Data transmission format in MASTER mode (cpha = ‘0’ cpol = ‘0’)

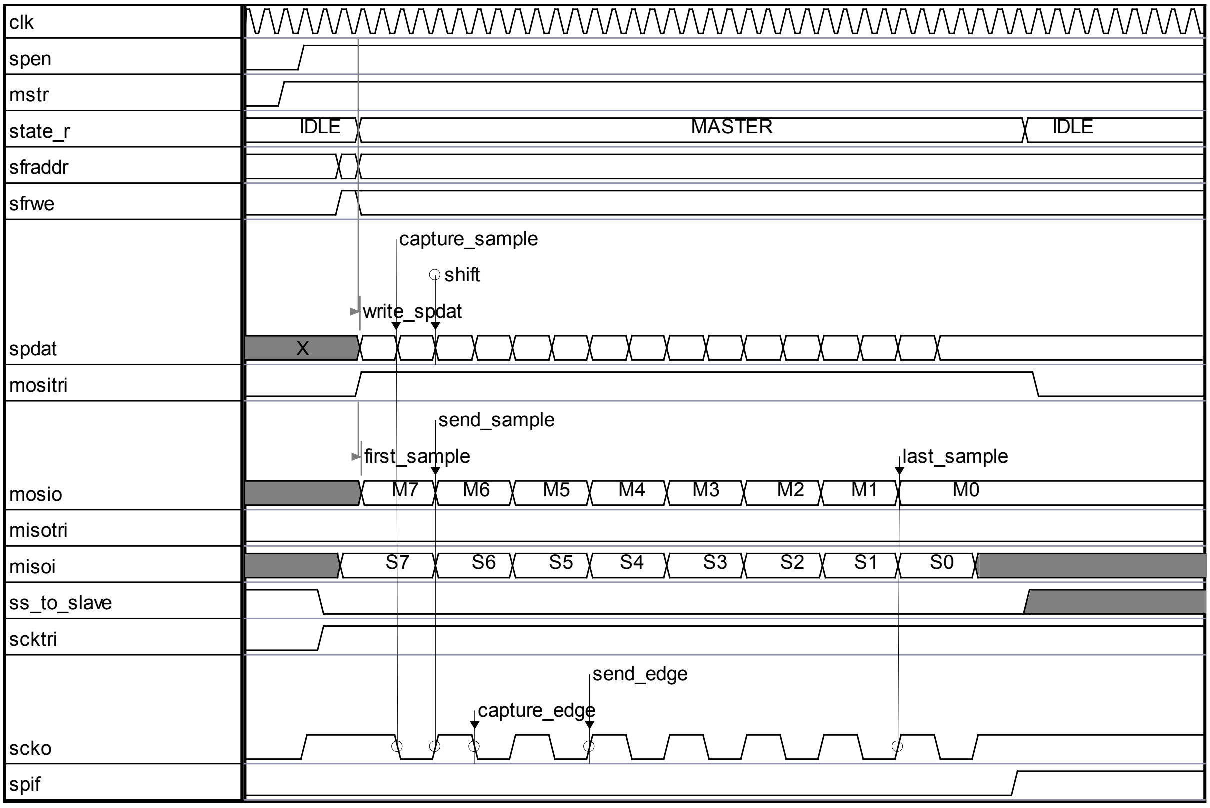


Figure 138. Data Transmission Format in MASTER Mode (cpha = '0' cpol = '1')

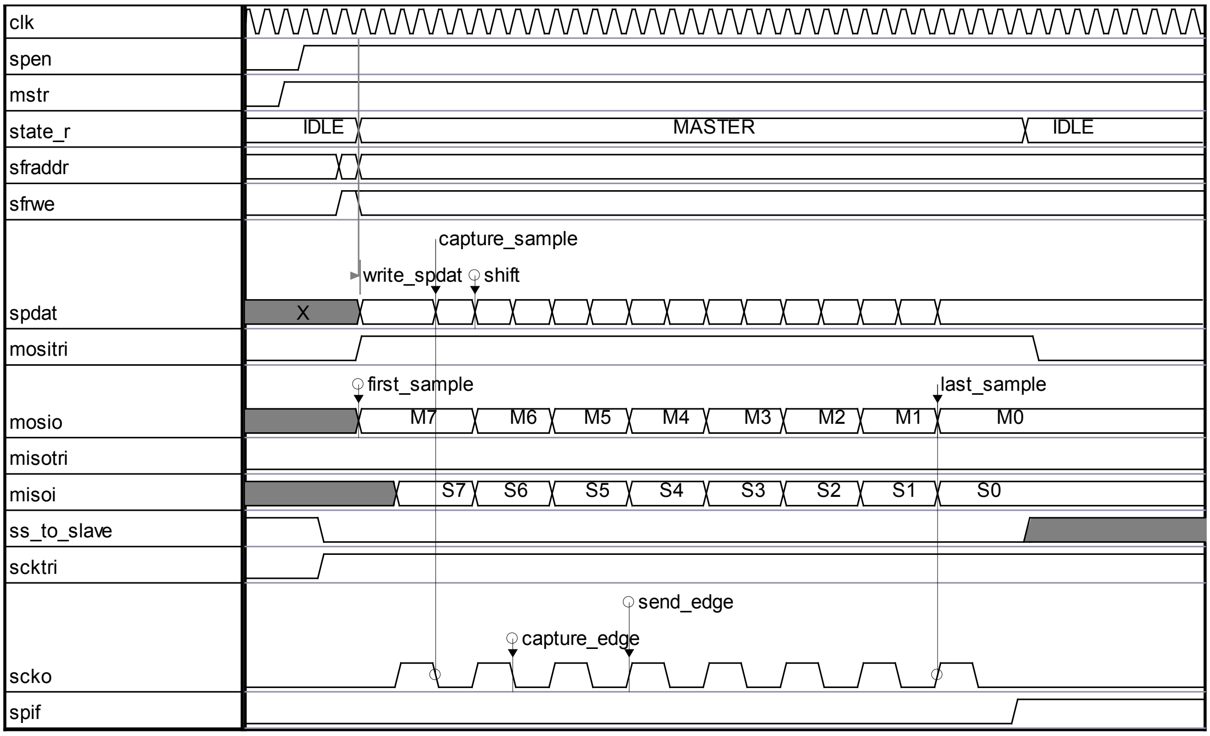


Figure 139. Data Transmission Format in MASTER Mode (cpha = '1' cpol = '0')

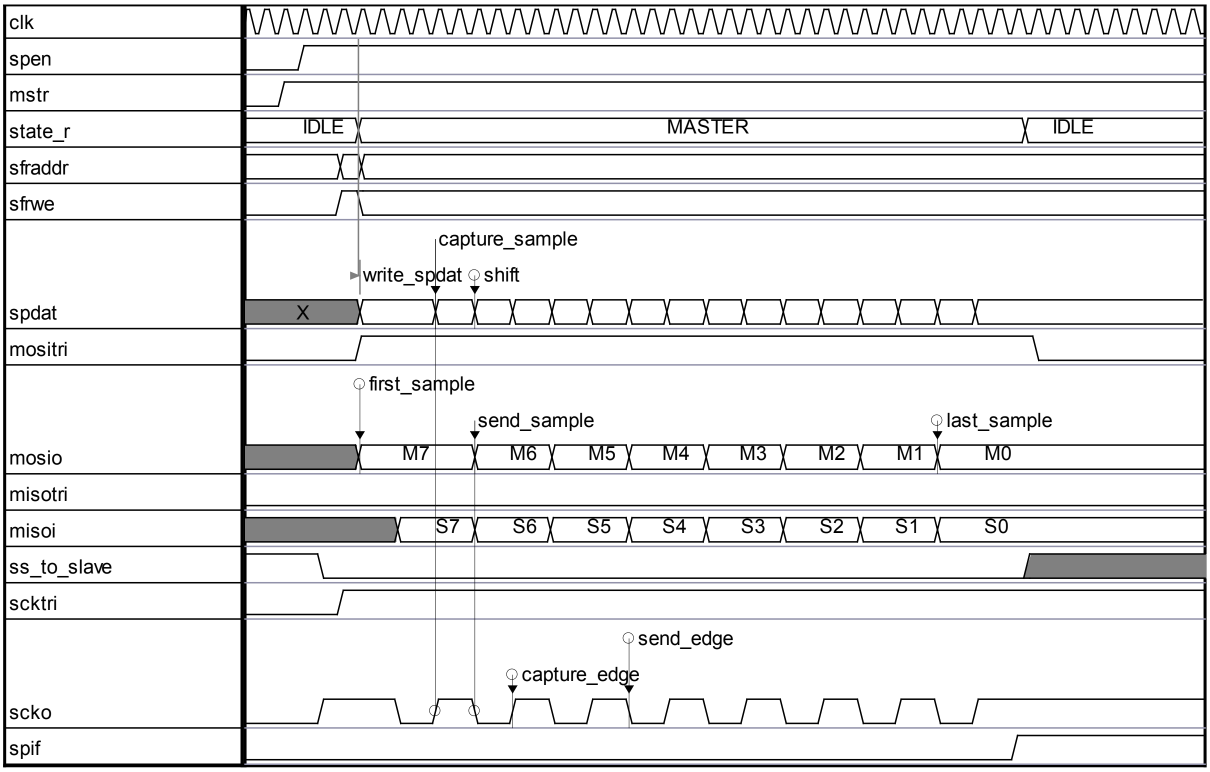


Figure 140. Data Transmission Format in MASTER Mode (cpha = '1' cpol = '1')

- SLAVE MODE

  First, the SPI_MS module has to be configured as a slave by writing "mstr"=0 in "spcon" register. Then it has to be enabled by setting the "spen"=1. Figure 141 presents process of data transmission in slave mode.

  The configuration : "cpha" = '1', "cpol" = '1', baud rate is f/4 (values of bits spcon(7), spcon(1), spcon(0) have no effect in this mode).

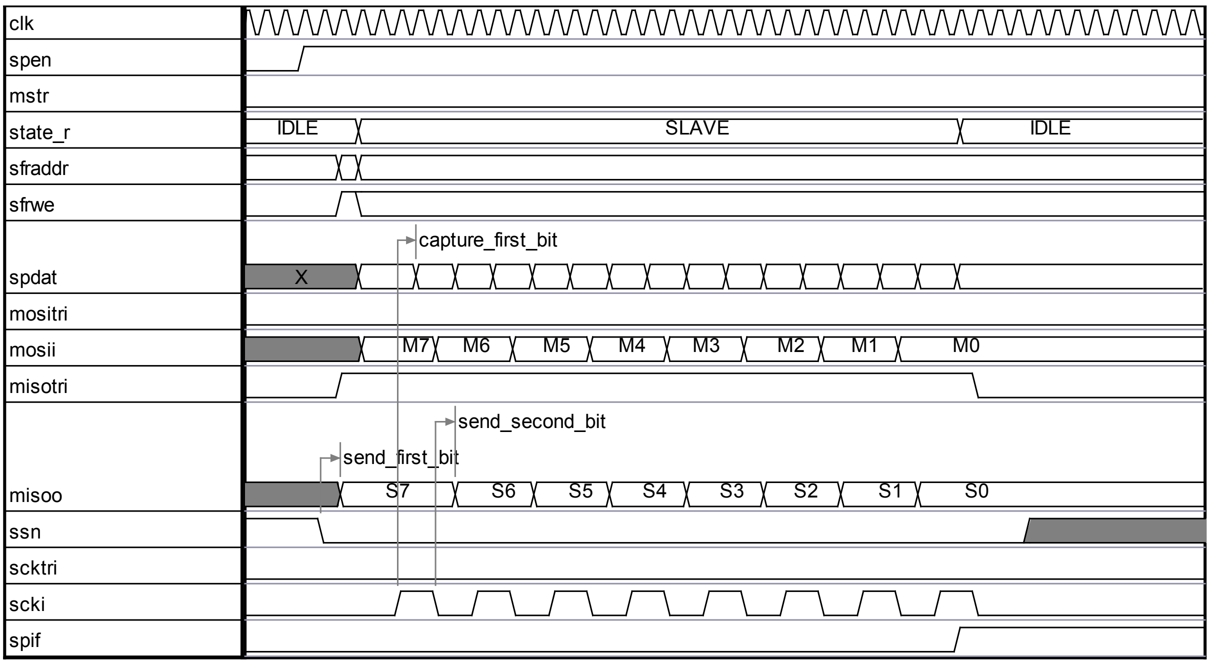


Figure 141. Data Transmission Format in SLAVE Mode

In slave mode the SPI_MS waits on low level on "ssn" input. The "ssn" input must remain low until the transmission is completed. The beginning of transmission depends on the state of the "cpha" bit of SPCON register (Figure 141, Figure 142). When "cpha" is cleared, then the slave must begin driving its data before the first "scki" edge, and a falling edge on the "ssn" input is used to start the transmission. When the "cpha" bit is set, then the slave uses the first edge of "scki" input as a transmission start signal.

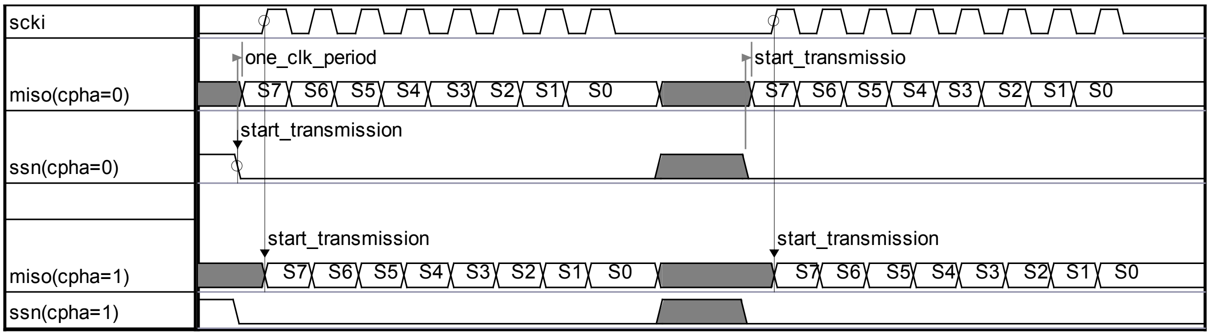


Figure 142. Data Transmission in SLAVE Mode For "cpha" = 0 and "cpha" = 1

c) Interrupt Generation

The SPI_MS provides the SPI Interrupt output "intspi". Two status flags can generate interrupt request (see Table 113).

Table 113. SPI_MS Interrupt Flags

Name	Description
"spif"	When the transmission is finished, this flag is set by hardware.
"modf"	This bit is set when the level on "ssn" input is in conflict with actual mode, i.e. it is '0' in Master mode (if "ssdis" = '0').

Figure 143 presents interrupt request caused by the "spif" flag. When the transmission ends the "spif" flag is set automatically by hardware.

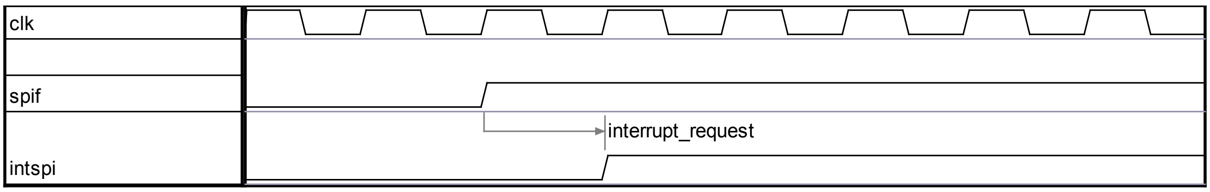


Figure 143. Interrupt Request Generated by "spif" Flag

Figure 144 presents interrupt request caused by "modf" flag. The "modf" flag is set automatically by hardware when level on the "ssn" input is inconsistent with respect to the selected operation mode (the SPI_MS module is configured in master mode and there is low level detected at "ssn" input) and the "ssdis" flag is cleared.

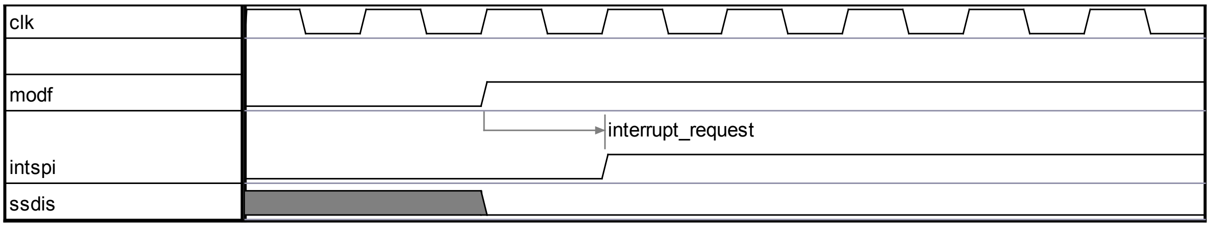


Figure 144. Interrupt Request Generated by "modf" Flag

Figure 145 presents situation when the interrupt request does not occur even there is "modf" condition detected. When the "ssdis" flag is set the "modf" interrupt is not generated.

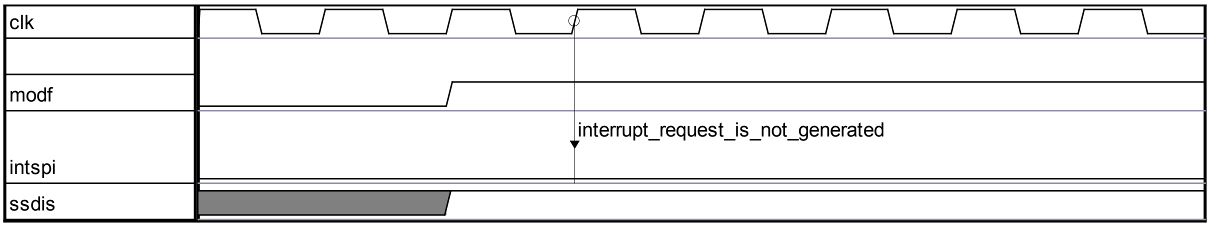


Figure 145. Interrupt Request Not Generated by "modf"

The interrupt request is disabled when both flags "spif" and "modf" are cleared.

The SPI interrupt shares the same vector as the External Interrupt 2. The "intspi" output signal is OR-ed with the External Interrupt 2 edge flag, before it comes into the Interrupt Controller (ISR). to determine the actual source of that interrupt, the "modf" and "spif" flags have to be investigated by the interrupt service routine. Since the External Interrupt 2 flag is automatically cleared after vectoring to the service subroutine, only the state of the "modf" and "spif" flags brings the information about the actual source of the interrupt.

5.20. OCDS

5.20.1 Overview

The OCDS implements functions which allow to stop/run/step the CPU. The OCDS is divided into two parts: first of them is a block responsible for JTAG communication, nexus operation control and control/status registers; second block controls debug mode entering, memory access functions and instruction substitution (step from OCDSINSTR register).

5.20.2 Block Diagram

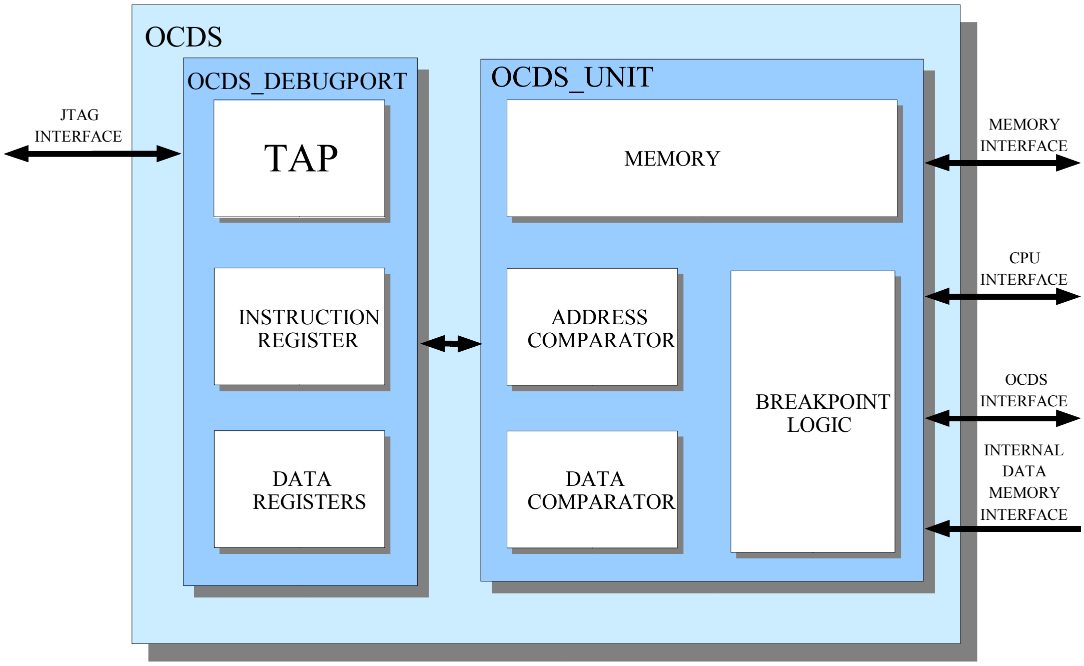


Figure 146. OCDS Block Diagram

- **TAP** – TAP module controls JTAG communication.
- **INSTRUCTION REGISTER** – the Instruction Register contains current JTAG instruction.
- **DATA REGISTERS** – Data Registers holds OCDS control and status bits.
- **MEMORY ACCESS** – the memory access module is used for memory read/write operations, single instruction executing (from code memory or OCDSINSTR register).
- **ADDRESS COMPARATOR** – the Address Comparator is used to determine if current read/write address is in the range selected in breakpoint address registers.
- **DATA COMPARATOR** – the Data Comparator is used to determine if current read/write data fulfill conditions selected in data and data mask breakpoint registers.
- **BREAKPOINT LOGIC** – the Breakpoint Logic Unit is responsible for generation of halting signals to the CPU.

5.20.3 Pin Description

Table 114. OCDS Pin Description

Name	Type	Polarity Bus size	Description
clk	I	High	**Clock** Pulse for all internal synchronous circuits in the OCDS_UNIT and the OCDS_TRACE components.
rst	I	High	**Reset input** External reset signal.
debugrst	O	High	**Reset output** Internal reset signal from OCDS.

Name	Type	Polarity Bus size	Description
tck	I	Rise	**Test Clock** Test clock signal (JTAG 1149.1). Clock pulse for all internal synchronous circuits in TCK clock domain (the OCDS_DEBUGPORT component).
tms	I	High	**Test mode select** Test mode signal (JTAG 1149).
tdi	I	High	**Test data input** Test data input (JTAG 1149.1).
trst	I	Low	**Test reset** Test reset signal (JTAG 1149).
tdo	O	High	**Test data output** Test data output (JTAG 1149.1).
tdoenable	O	High	**External tristate buffer** Control signal for external tdo tristate buffer.
debugack	I	High	**Debug mode acknowledge** Debug acknowledge signal from CPU.
flush	I	High	**Flush** Program branch indicator.
codefetche	I	High	**Code fetch** Code fetch indicator.
datafetche	I	High	**Data fetch** Data fetch indicator.
lastcycle	I	High	**Last cycle of instruction** Last cycle of instruction indicator.
allfetch	I	High	**All instruction fetch** All instruction fetch (performed and annuled indicator.
acc	I	OCDSACC_LEN	**Accumulator register input** Accumulator contents.
dr0	I	OCDSDR0_LEN	**DR0 register input** DR0 contents.
pcreg	I	OCDSPC_LEN	**Program Counter register input** Program counter contents.
debugreq	O	High	**Debug mode request** Debug mode request signal to CPU.
debugstep	O	High	**Debug step** Debug step signal (single instruction execution).
debugprog	O	High	**Debugger program** Instruction fetch source select signal.
datao	I	8	**External memory data bus input** Memory data bus output from CPU, connected to external data memory input. Combinatorial input signal for OCDS used for breakpoint detection.

Name	Type	Polarity Bus size	Description
memaddr	I	24	**External memory address bus** External memory address bus output from CPU, connected to external data memory input. Combinatorial input signal for OCDS used for breakpoint detection.
memwr	I	High	**External memory write enable** External memory write signal output from CPU (memwr), connected to OCDS input. Combinatorial input signal.
memrd	I	High	**External memory read enable** External memory read signal output from CPU (memrd), connected to OCDS input. Combinatorial input signal.
waitstaten	I	Low	**Waitstate indicator** Active low when the CPU performs a wait cycle (internally or externally generated)
memdatai	O	8	**CPU memory data bus input** Memory data bus output from OCDS, connected to CPU (memdatai input port). Combinatorial output signal driven with either external data memory output bus or substituted instruction (in step mode or memory access).
debugmemdatai	I	8	**External memory data bus output** Memory data bus output from external memory, connected to OCDS input. Combinatorial input signal.
ramdatai	I	8	**Internal RAM data output** Internal RAM data output bus (from internal RAM to processor) connected to OCDS for breakpoint detection. Combinatorial input signal.
ramaddr	I	8	**Internal RAM address bus** Internal RAM address bus connected to OCDS for breakpoint detection. Combinatorial input signal.
ramwe	I	High	**Internal RAM write enable** Internal RAM write enable signal connected to OCDS for breakpoint detection. Combinatorial input signal.
ramoe	I	High	**Internal RAM output enable** Internal RAM read enable signal connected to OCDS for breakpoint detection. Combinatorial input signal.
Outputs designed for PMU (when PMU_IMPLEMENT = 1 only)			
debugpmureq	O	High	**Debug request for PMU** Debug request for PMU – wake up from IDLE or STOP mode when debug mode is entered (the dreq bit in OCDSCTRL register is set).

Name	Type	Polarity Bus size	Description
debugperen	O	High	**Peripherals clock enable** Peripheral clock enable signal (when inactive – peripherals doesn't work).
Output designed for Program Memory Write			
debugpmw	O	High	**OCDS program memory write** Program memory write signal for debugger using (memory write).
Inputs and outputs designed for Trace (when OCDS_TYPE parameter is 1 or 2 only)			
addr_buf0	O	BUF_SIZE	**RAM0 address** RAM0 address bus.
datao_buf0	O	BUF_LEN	**RAM0 data output** RAM0 data output bus.
datai_buf0	I	BUF_LEN	**RAM0 data input** RAM0 data input bus.
wr_buf0	O	High	**RAM0 write enable** RAM0 write enable signal.
rd_buf0	O	High	**RAM0 read enable** RAM0 read enable signal.
addr_buf1	O	BUF_SIZE	**RAM1 address** RAM1 address bus.
datao_buf1	O	BUF_LEN	**RAM1 data output** RAM1 data output bus.
datai_buf1	I	BUF_LEN	**RAM1 data input** RAM1 data input bus.
wr_buf1	O	High	**RAM1 write enable** RAM1 write enable signal.
rd_buf1	O	High	**RAM1 read enable** RAM1 read enable signal.

•

5.20.4 Functional Description

a) JTAG Instruction Register

This section presents implementation of the JTAG instruction register.

The instruction register is 4 bit long. All the instructions that are implemented are summarized in Table 115.

Most of the debug registers are accessible according to the way described in the Nexus 5001 standard (except for memory access register OCDSMEMACC register). All the NEXUS registers are accessible through the IEEE 1149.1 port independently of the state of the target processor.

Table 115 Implemented Instructions

Code	Name	Chosen DR (Source of data)	Description

Code	Name	Chosen DR (Source of data)	Description
0001	ID-CODE	ID Register	32 bit ID register (the value can by defined by user before core synthesis). Required by IEEE standard.
1001	BUFF	Trace Buffer Register	Register containing trace frame during reading trace buffer. Instruction is implemented if configuration with trace is chosen.
1010	OCDSMEMACC	Memory Access Data Register	Data read from or that are to be written to memory.
1011	NEXUS-ENABLE	Nexus Registers	Refer to NRR selection 5.20.4 b).
1111	BYPASS	Bypass Register	Required by IEEE standard.
All others	BYPASS	Bypass Register	Some codes can be used in a future versions. It is recommended not to use them.

b) NRR Access

Access to the NRRs is enabled when the IEEE 1149.1 controller decodes a device-specific NEXUS-ENABLE instruction entered via the SELECT_IR_SCAN path. When the IEEE 1149.1 controller passes through the UPDATE-IR state and decodes the NEXUS-ENABLE instruction, the Nexus controller will be reset to the NRR select state. The Nexus controller can work in one of the three states: idle, register select state and register data access state. Table 116 illustrates the IEEE 1149.1 sequence of selection of the Nexus controller.

Table 116. The IEEE 1149.1 Sequence to Enable Nexus Block For Communication

Step	TMS	IEEE 1149.1 State	Nexus State	Description
1	0	RUN_TEST_IDLE	IDLE	IEEE 1149.1 controller in reset state.
2	1	SELECT_DR_SCAN	IDLE	
3	1	SELECT_IR_SCAN	IDLE	
4	0	CAPTURE_IR	IDLE	Load value "0001" into TDI/TDO shifter[1].
5	0	SHIFT_IR	IDLE	TDO becomes active and the IEEE 1149.1 shifter is ready. Shift 3 bits of size of vendor-defined Nexus-Enable Instruction.
3 TCKs				
6	1	EXIT1_IR	IDLE	Last bit of Device ID shifted out to TDO.
7	1	UPDATE_IR	REG_SELECT	IEEE 1149.1 controller decoder. Nexus controller is forced to register select state.
8	0	RUN_TEST_IDLE	REG_SELECT	Nexus controller enabled and ready to receive commands.

When the IEEE 1149.1 NEXUS-ENABLE instruction is decoded by the IEEE 1149.1 controller, the IEEE 1149.1 port allows tool/target communication with up to 128 IEEE 1149.1 NRRs. Each NRR is referenced by a unique register address index in the range 0 through 127.

[1] . IEEE 1149.1 Specifies that IR should capture the value XX..X01. Nexus 5001 is in this point inconsistent with IEEE1149.1

All the communication with the Nexus controller is performed via the SELECT_DR_SCAN path. The Nexus controller will default to a register select state when enabled. Accessing an NRR requires two passes through the SELECT_DR_SCAN path: one pass to select the NRR and the second one to read/write the NRR.

The first pass through the SELECT_DR_SCAN path is used to enter an 8-bit Nexus command which consists of a read/write control bit in the LSB followed by a 7-bit NRR address, as it is illustrated in the Table 117.

Table 117. The Ieee 1149.1 Controller Command Input

Bits 7..1	Bit 0
7 bit NRR address	R/W
Note: R/W=0 – write R/W=1 – read	

The second pass through the SELECT_DR_SCAN path is used to read or write the NRR data by shifting in the data LSB first during the SHIFT_DR state. When an NRR is read, the register value is loaded into the IEEE 1149.1 shifter during the CAPTURE_DR state. When an NRR is written, the value is loaded by the IEEE 1149.1 shifter to the NRR during the UPDATE_DR state. While reading data from an NRR, there is no requirement to shift out the entire NRR content, and shifting may be terminated once the required number of bits have been acquired. Figure 147 illustrates the relationship between an IEEE 1149.1 TAP state machine and a Nexus controller state machine.

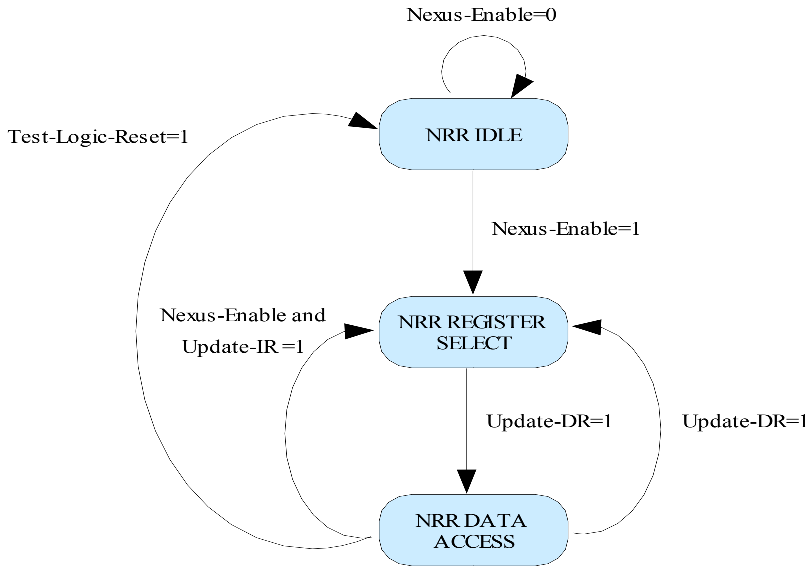


Figure 147. The IEEE 1149.1 TAP State Machine vs NEXUS State Machine

Table 118 OCDS Special Function Registers

Step	TMS	IEEE 1149.1 State	Nexus State	Description
1	0	RUN_TEST_IDLE	NRR REG_SELECT	IEEE 1149.1 controller in reset state.
2	1	SELECT_DR_SCAN	NRR REG_SELECT	
3	0	CAPTURE_DR	NRR REG_SELECT	IEEE 1149.1 shifter may be loaded with last value of register being decoded by Nexus controller or Nexus status information.
4	0	SHIFT_DR	NRR REG_SELECT	TDO becomes active and NRR address and write bit is shifted in.

Step	TMS	IEEE 1149.1 State	Nexus State	Description
7 TCKs			NRR REG_SELECT	
5	1	EXIT1_DR	NRR REG_SELECT	Last bit of NRR shifted into TDI.
6	1	UPDATE_DR	NRR REG_SELECT	Nexus controller decodes and selects register.
7	1	SELECT_DR_SCAN	NRR DATA ACCESS	Second pass through SELECT_DR_SCAN
8	0	CAPTURE_DR	NRR DATA ACCESS	IEEE 1149.1 shifter may be loaded with last value of register being decoded by Nexus controller or Nexus status information.
9	0	SHIFT_DR NRR	DATA ACCESS	TDO becomes active and outputs current value of register while new value is shifted in through TDI.
N-1 TCKs			NRR DATA ACCESS	
10	1	EXIT1_DR	NRR DATA ACCESS	Last bit of NRR shifted out to TDO.
11	1	UPDATE_DR	NRR DATA ACCESS	Nexus controller writes value to register.
12	0	RUN_TEST_IDLE	NRR REG_SELECT	IEEE 1149.1 controller returns to idle state, or may return to SELECT_DR_SCAN state for new NRR register select. Total number of TCK clocks = 49 in this example.

c) Jtag Interface

An IEEE 1149.1 port used for this standard implements all the mandatory features of the standard IEEE 1149.1 port, including the "BYPASS" and "IDCODE" instructions. A 16-state IEEE 1149.1 TAP state machine is used per the IEEE-1149.1 standard as illustrated in Figure 148.

The 5 required IEEE 1149.1 pins are as follows:

- Test Data Input (TDI) provides for serial movement of data into the IEEE 1149.1 port.
- Test Data Output (TDO) provides for serial movement of data out of the IEEE 1149.1 port. All target accesses initiated via the IEEE 1149.1 port are transmitted by the target via TDO
- Test Clock Input (TCK) provides the clock for the IEEE 1149.1 port.
- Test Mode Select Input (TMS) provides access to the IEEE 1149.1 TAP state machine.
- Test Reset Input (TRST) provides for asynchronous initialization of the IEEE 1149.1 controller. TRST signal is implemented in the model and it is core's user responsibility to provide reset signal on power on to guarantee that debug feature of embedded processor core will be disabled on power on. Optionally (it depends on the user), the TRST signal can be accessed as an external pin. The signal levels for the five IEEE 1149.1 pins are defined by the hardware in which the core is implemented.
- The timing for the five IEEE 1149.1 pins is in agreement with the IEEE 1149.1 standard[2].

[2] I.e. any JTAG-probe that adheres to the IEEE 1149.1 specification can be used to control the OCDS

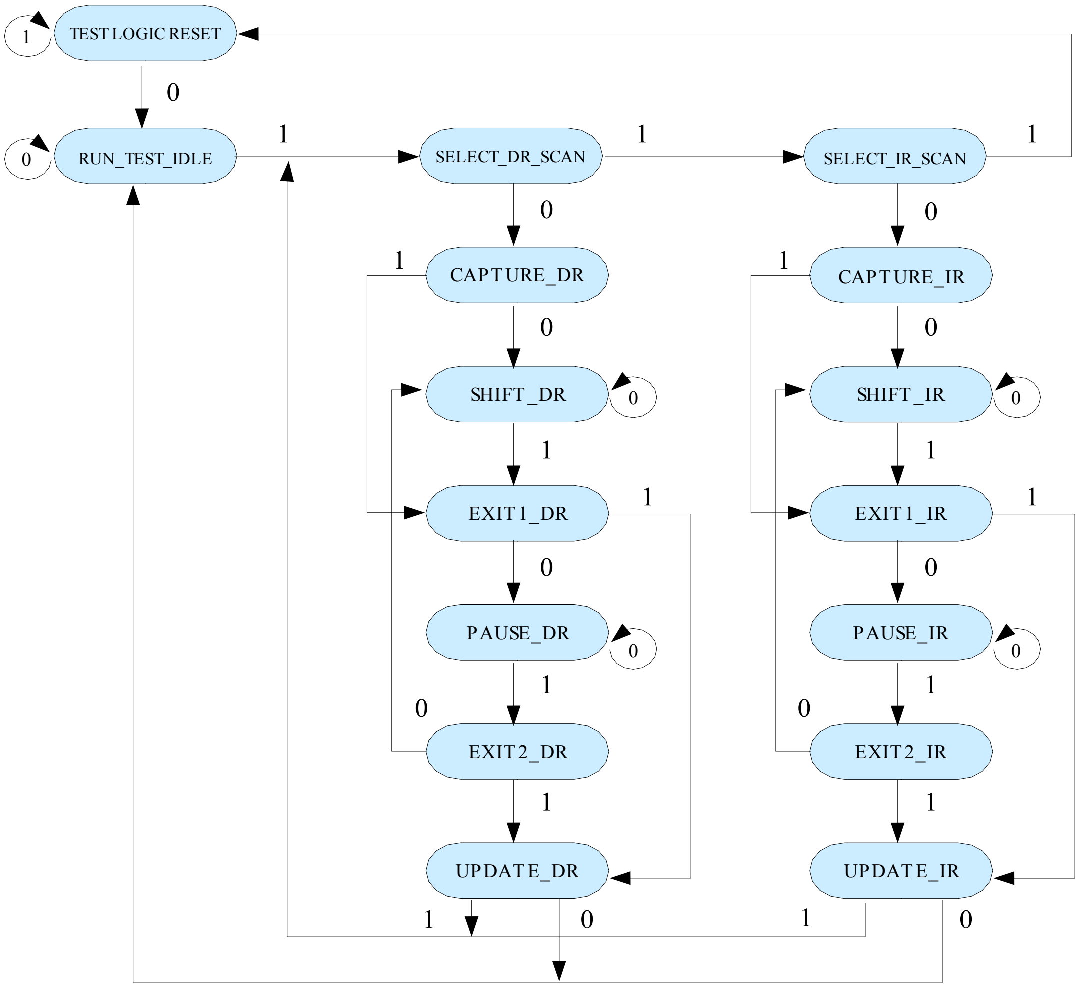


Figure 148. The Ieee 1149.1 Tap State Machine

Assertion of a power on reset signal on the embedded processor or the TRST pin causes the IEEE 1149.1 controller to default to being loaded with the IDCODE instruction upon exit of TEST_LOGIC_RESET controller state. This allows for immediate entry to the SELECT_DR_SCAN path to retrieve the contents of the device ID. The contents of ID Code register can be modified by user however the LSB of the IDCODE must be a logic 1 so that examination of the first bit of data, shifted out of a component during a data scan sequence immediately following exit from the TEST_LOGIC_RESET controller state, shows whether a Device Identification (DID) Register is included in the design, as required by Nexus 5001 standard.

The core system logic will continue its normal operation undisturbed when the IEEE 1149.1 controller decodes the IDCODE instruction.

d) Deviation From the IEEE 1149.1 Standard

The synthesizable core model is targeted mainly on FPGA. For optimisation reasons the following limitation was applied:

- Only one global clock network is used for all FF’s in the whole design interface
- Only FF’s are used (there is no latches)
- The vast majority of FF’s work on rising edge of clock (only 4 FF’s work on falling edge)

- Only limited number of FF's uses asynchronous reset - TAP FSM Flip-Flops, Instruction Register and OCDSCTRL register. All the others are reset synchronously.
- For portability reason no device specific features (e.g. XILINX FPGA's provide internal TAP) are used in the interface.

- The reason mentioned above leads to the following deviation from the IEEE standard:
  - All registers (IR, DRs) are updated at rising edge of TCK while exiting UPDATE state.
  - Assertion of asynchronous TRST signal sets only TAP-FSM, Instruction Register and OCDSCTRL register to initial state, what is enough for core system logic to continue its normal operation undisturbed. All the other registers are reset when TAP FSM is in TEST_LOGIC_RESET state at first rising edge of TCK.
  - EXTEST, INTEST and other instructions that refer to Boundary Scan Path (BSP) are not implemented (even if BSP is implemented in FPGA it is device specific feature and is not used).

e) JTAG Timing

The exemplary waveforms show the following operations:

- Test Logic Reset - performed without using TRST signal (TMS=1, 5 TCP pulses)
- Writing Nexus Enable Code into IEEE TAP Instruction Register (Nexus enable code 1101)
- Writing value 008054h to OCDSBPAM1 (23 bit) register (OCDSBPAM1 - address 89h)

Signal description:

tck, tms, tdi, tdo	- IEEE 1149.1 Test Access Port pins.
tdoenable	- output signal
ins	- contents of the instruction register (internal)
nxaddress	- Nexus address (internal)
ocdsbpam1	- contents of the OCDSBPAM1 register (internal)
tapstate	- TAP controller state (internal). For codes[3] refer to Table 119.

Table 119 TAP States Codes

TAP STATE	CODE (hex)
TEST_LOGIC_RESET	F
RUN_TEST_IDLE	C
SELECT_DR_SCAN	7
CAPTURE_DR	6
SHIFT_DR	2
EXIT1_DR	1
PAUSE_DR	3
EXIT2_DR	0
UPADATE_DR	5
SELECT_IR_SCAN	4
CAPTURE_IR	E

[3] the State Codes are chosen according to IEEE149.1 appendix on exemplary implementation

TAP STATE	CODE (hex)
SHIFT_IR	A
EXIT1_IR	9
PAUSE_IR	B
EXIT2_IR	8
UPADATE_IR	D

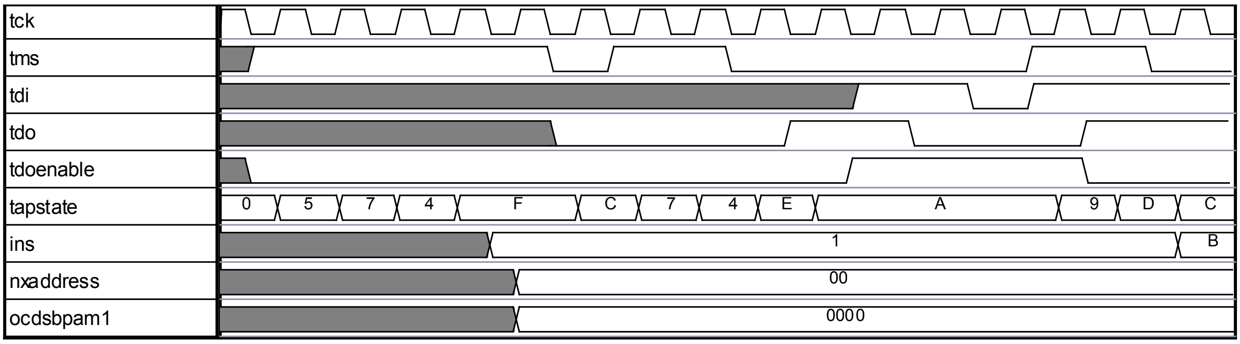


Figure 149. Reset and the NEXUS_ENABLE Instruction Write

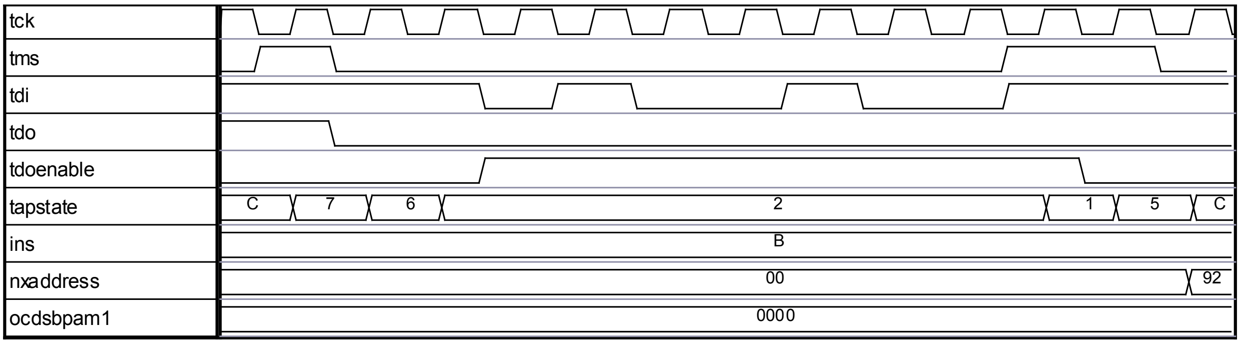


Figure 150. The OCDSBPAM1 Register Select

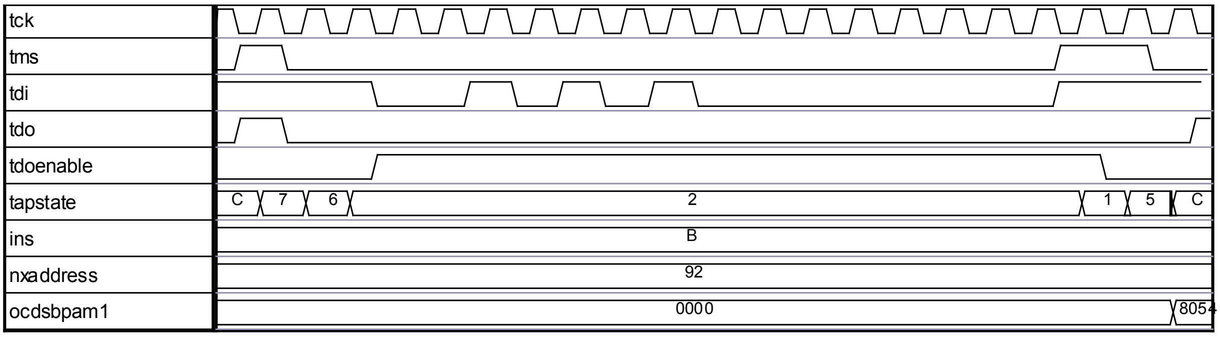


Figure 151. The OCDSBPAM1 Write

5.20.5 Trace Data Register (BUFF)

The Trace Data Register (BUFF) is implemented when OCDS_TYPE parameter value is 1 or 2 during IP core configuration.

Table 120. BUFF Register

Bit	Number	Description

Bit	Number	Description
Empty	n	Empty status bit empty ='1' – data trace buffer is empty and frame field is not valid trace frame empty ='0' – data trace buffer is not empty and frame field is valid trace frame
Frame	n-1..0	Trace frame from trace buffer.

Trace buffer read procedure:

- Enter debug mode.
- Write BUFF instruction into the JTAG instruction register.
- Read BUFF data register through the data scan.

For more details see 5.20.11 .

5.20.6 Special Function Registers

Table 121 OCDS Special Function Registers

Register	Nexus location (decimal)	Reset value	Length	Read/ Write	Description
OCDSCTRL	64	00H	11	RS/W	Control register
OCDSACC	65	See Note 3	8	RO	Accumulator
OCDSPC	66	See Note 3	24	RO	Program Counter
OCDSINSTR	67	000000H	32	RW	Instruction Substitution
OCDSMAC	68	00H	8	RW	Memory Access Control
OCDSTRC	69	0H	4	RS/W	Trace control register
OCDSDR0	70	00000000H	32	RO	DR0 register
OCDSBPD1	71	00H	8	RW	Breakpoint 1 Data
OCDSBPDM1	72	00H	8	RW	Breakpoint 1 Data Mask
OCDSBPA1	73	000000H	24	RW	Breakpoint 1 Start Address
OCDSBPAM1	74	000000H	24	RW	Breakpoint 1 End Address
OCDSBPC1	75	00H	8	RS/W	Breakpoint 1 Control
OCDSBPD2	76	00H	8	RW	Breakpoint 2 Data
OCDSBPDM2	77	00H	8	RW	Breakpoint 2 Data Mask
OCDSBPA2	78	00000H	24	RW	Breakpoint 2 Start Address
OCDSBPAM2	79	00000H	24	RW	Breakpoint 2 End Address
OCDSBPC2	80	00H	8	RS/W	Breakpoint 2 Control
OCDSBPD_N	71+5*(N-1)	00H	8	RW	Breakpoint N Data
OCDSBPDM_N	72+5*(N-1)	00H	8	RW	Breakpoint N Data Mask
OCDSBPA_N	73+5*(N-1)	000000H	24	RW	Breakpoint N Start Address
OCDSBPAM_N	74+5*(N-1)	000000H	24	RW	Breakpoint N End Address
OCDSBPC_N	75+5*(N-1)	00H	8	RS/W	Breakpoint N Control

Register	Nexus location (decimal)	Reset value	Length	Read/ Write	Description
Read/Write Access description: *RO - Read Only* *RW - When Read the last written value is returned* *RS/W - Written value is used for control, when read returns some status information*					
Note 1: Write of the program counter register is done by the LJMP #addr instruction.					
Note 2: Registers OCDSACC, OCDSPC can be read in any time and this operation does not cause any interference for the core. However if processor is running the values read from the above register are unpredictable because there is no synchronization mechanism between core and these registers, accessed via IEEE 1149.1.					
Note 3: Physically OCDSACC and OCDSPC registers does not exist. When they are read current accumulator and program counter contents are always returned.					

a) OCDSCTRL Register

Table 122 OCDSCTRL Register

Bit	Number	Description
swbstop	10	Software breakpoint status swbstop ='1' software breakpoint occurred swbstop ='0' software breakpoint did not occur
-	9	Not used.
dprog1	8	Debugger program. Program selector 1: dprog1='1' – (and if dprog0='1') instruction is fetched from OCDSINSTR register, but Program Counter (PC) is normally incremented. If the fetched instruction is a jump instruction, PC will be reloaded according to the argument provided by OCDSINSTR. dprog1='0' – user program, instruction is fetched from the program memory, PC is normally incremented (except when dprog0='1').
-	7	Not used.
clkperen	6	Peripheral clock enable clkperen='1' – peripheral's clock is enabled in the debug mode clkperen='0' - peripheral's clock is disabled in the debug mode
rst	5	Reset The rst bit is connected directly to the microcontroller's reset input. Notice that setting of the core reset active causes that no other operation on core can be performed, e.g. reading or writing memory, loading registers etc.
ocdsen	4	OCDS enable ocdsen='1' - OCDS is enabled ocdsen='0' - OCDS is disabled. The microcontroller is not stopped at the breakpoints. 0xA5A5 instruction is executed like NOP instruction. The rst bit is ignored.

Bit	Number	Description
dack	3	Debug acknowledge dack='1' – the microcontroller is stopped (at the breakpoint, at the software breakpoint or external debug request dreq='1') dack='0' – the microcontroller executes instructions. Value written to this bit is ignored.
dreq	2	Debug request external input dreq='1' – the microcontroller enters debug mode and asserts dack='1'. dreq='0' – the microcontroller leaves debug mode and asserts dack='0'. dreq is also used to start execution of user program after reaching breakpoint. in such case dreq bit must be set and then cleared.
dstep	1	Debug step When dstep is set to '1' it causes processor to execute single instruction. After that, this bit is automatically cleared. So writing '1' again causes execution of the following instruction. When read always equal to '0'. NOTE: It is **not allowed** to set DREQ bit and DSTEP bit simultaneously (if DREQ bit was previously equal to '0') For synchronization reason the result of such operation is unpredictable.
dprog0	0	Debugger program Program selector 0: dprog0='1' – debugger program, instruction is fetched from OCDSINSTR register. Program Counter (PC) is not incremented then. However if the fetched instruction is a jump instruction, PC will be reloaded according to the argument provided by OCDSINSTR. dprog0='0' – user program, instruction is fetched from the program memory (except when dprog1='1'). PC is normally incremented.

b) OCDSCACC Register

Table 123 OCDSACC Register

Bit	Number	Description
acc	7..0	Accumulator contents

c) OCDSCDR0 Register

Table 124 OCDSDR0 Register

Bit	Number	Description
dr0	31..0	DR0 register contents

d) OCDSPC Register

Table 125 OCDSPC Register

Bit	Number	Description
pc	23..0	Program counter contents

e) OCDSINSTR Register

Table 126 OCDSINSTR Register

Bit	Number	Description
ocdsinstr3	31..24	The instruction register
ocdsinstr2	23..16	The instruction register 1
ocdsinstr1	15..8	The argument register 2
ocdsinstr0	7..0	The argument register 3

f) OCDSMAC Register

Table 127 OCDSMAC Register

Bit	Number	Description
regfilesel	3	Register File/Memory select regfilesel='1' - Read/write of the Register File regfilesel='0' - Read/write of program/data memory
memrd	2	Memory read memrd='1' - Read of the program/data memory memrd='0' - Normal mode
memwr	1	Memory write memwr='1' - Write of the program/data memory memwr='0' - Normal mode
daprosel	0	Data/program memory select daprosel='1' - Read/write of the data memory daprosel='0' - Read/write of the program memory

Reading from and writing to memory is performed by causing core to execute the sequence of proper instructions. For that reason reset signal provided to the core must be '0' (rst bit in OCDSCTRL register must be equal to '0' and external reset signal must be equal to '0), and processor must be stopped (OCDSCTRL - dreq = '1').

Performing read from/write to memory requires reloading DPTR register first and change the state of accumulator. For that reason ACC and DPTR must be saved earlier and restored after finishing of the operation.

For read/write of the Register File, the 'regfilesel' bit must be set. The value of the ACC[3:0] register prior to setting the 'memrd' or 'memwr' bit will be used as pointer to the Register File double word (ACC=0 points to DR0, ACC=15 points to DR60). The 32-bit values written to OCDSMEMACC register will be written to the DR0 and then to the selected Register File 'double word'; the values read from the selected 'double word' register in the Register File will be passed through the DR0 to the OCDSMEMACC register. For that reason the ACC and DR0 must be stored earlier and restored after finishing of the operation.

g) OCDSTRC Register

The Trace Control Register (OCDSTRC) is implemented when OCDS_TYPE parameter value is 1 or 2 during IP core configuration.

Table 128. OCDSTRC Register

Bit	Number	Description
switch	3	Switch trace bit. Bit is taken into the consideration common with *cont* bit and events starting trace – external trigger and writing to cont *bit* (start/end trace events configured in OCDSBP* registers are independent). Possible cases: Start event and switch ='1' – start program and data trace. Start event and switch ='0' – start program trace. Bit is always read as 0 in configuration without trace option.
cont	2	Continue bit. Leaving the debug mode with *cont* ='0' ends trace (program and program and data). Leaving the debug mode with *cont* ='1' continues or starts trace (program and data). Setting *cont* bit ('0' to '1') in normal program execution mode starts trace (program and data). Bit is always read as 0 in configuration without trace option.
full	1	Trace buffer full status full ='1' trace buffer is full full ='0' trace buffer is not full Bit is always read as 0 in configuration without trace option.
full_bp_en	0	Breakpoint from trace buffer (on reaching full status) full_bp_en ='1' breakpoint enabled full_bp_en ='0' breakpoint disabled Bit is always read as 0 in configuration without trace option.

h) OCDSBPD Register

Table 129 OCDSBPD Register

Bit	Number	Description
ocdsbpd	7..0	Data value

i) OCDSBPDM Register

Table 130 OCDSBPDM Register

Bit	Number	Description
ocdsbpdm	7..0	Data mask value

j) OCDSBPA Register

Table 131 OCDSBA Register

Bit	Number	Description
ocdsbpa	23..0	Start address

k) OCDSBPAM Register

Table 132 OCDSBAM Register

Bit	Number	Description
ocdsbpam	23..0	End address

l) OCDSBPC Register

Table 133 OCDSBPC Register

Bit	Number	Description
mod	7..5	Breakpoint logic mode 000 – breakpoint 001 – start program trace 010 – end program trace 011 – start program and data trace 100 – end program and data trace 101 – start data trace 110 – end data trace 111 – data trace mode In configuration without any trace option only 000 value is allowed. In configuration with program trace only 000, 001 and 010 values are allowed. In configuration with full trace (program and data) all values are allowed. Writing to *mod* field not allowed value is treated as writing 000 value.
mss	4	Data memory select mssn='1' - internal data memory for breakpoint n mssn='0' - external data memory for breakpoint n
bpstop	3	Program stopped at the breakpoint bpstopn='1' - program stopped at the breakpoint "n" bpstopn='0' - breakpoint "n" inactive
memsel	2	Memory select memseln='1' - program memory select for breakpoint n memseln='0' - data memory select for breakpoint n
rsel	1	Read select rseln='1' - read accesses are monitored for breakpoint n rseln='0' - read accesses are not monitored for breakpoint n
wsel	0	Write select wseln='1' - write accesses are monitored for breakpoint n wseln='0' - write accesses are not monitored for breakpoint n

5.20.7 Clock

a) Clock Inputs

Table 134 Clock Inputs

Clock	Polarity	Clock description
clk	Rise	**Clock:** the CPU clock signal used in breakpoint conditions check
tck	Rise	**Test clock:** test clock signal (JTAG 1149.1)

b) Clock Desription

Clock clk is used for internal OCDS operations. All CPU signals are generated in the clk clock domain and internal OCDS logic uses them for halt conditions detection and control the CPU.

Clock tck is used for JTAG communication with host controller. All OCDS functions are accessed by writing or reading OCDS registers according to JTAG protocol.

Clock tck period must be at least two times greater then clk clock period.

c) Clock Domains

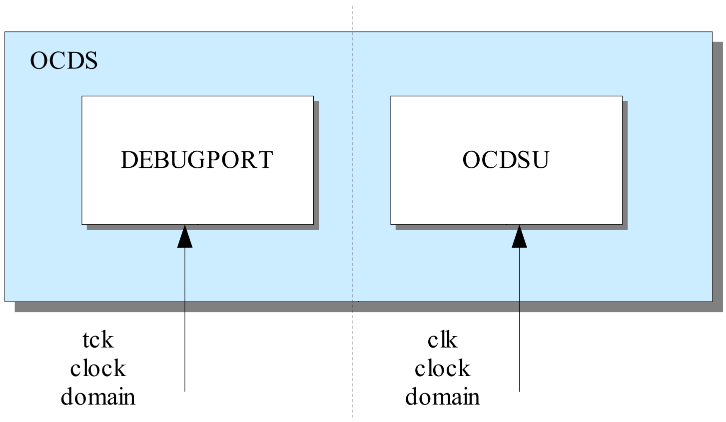


Figure 152. Clock Domains Diagram

The OCDS_DEBUGPORT unit is placed in the tck clock domain. All the registers in this domain are rising tck clock edge sensitive except for these ones (sensitive on the falling tck clock edge):SHIFTIR, SHIFTDR, RESET, ENABLE, TDO.

The OCDS_UNIT is placed in clk clock domain. All the registers in this domain are rising clk clock edge sensitive.

5.20.8 Reset

a) Reset Description

The OCDS core has two reset input signals: rst and trst.

The rst signal is reset signal (rstout) for all the flip-flops in clk clock domain – components: OCDS_UNIT and OCDS_TRACE.

The trst signal (in the OCDS_DEBUGPORT) is an asynchronous reset for TAP controller (refer to Figure 154). If trst is active TAP enters into TEST_LOGIC_RESET state and then on the first falling tck edge internal reset (reset) signal is activated. The trst signal asynchronously resets SHIFTIR, SHIFTDR, resets and enables flip-flops. Internal reset signal synchronously resets all the remaining registers, at the rising tck clock edge, except UP_IREG (instruction register) and

UP_REG_OCTRL (ocds control register) which are reset asynchronously. Internal reset signal resets tdo flip-flop on the falling tck clock edge.

The debugrst reset signal (for CPU) is generated on the basis two bits in OCDSCTRL register: OCDSEN and RST (see figure below).

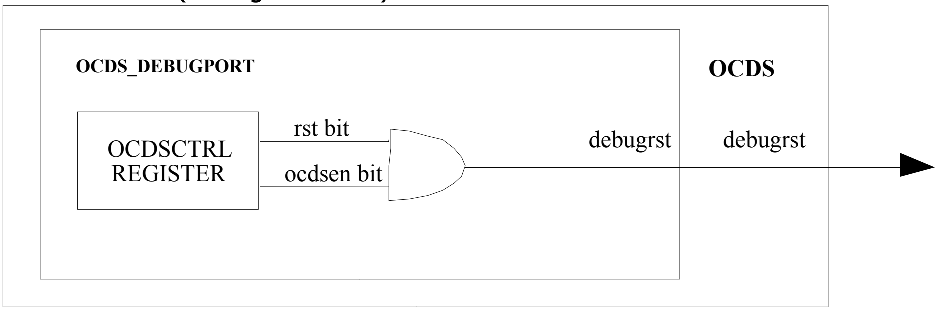


Figure 153. The Debugrst Reset Signal Generation

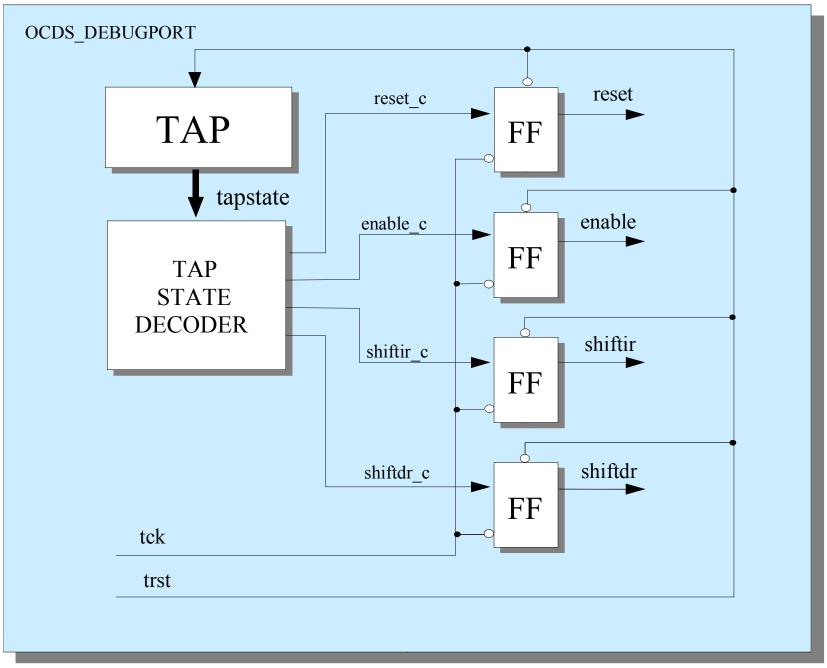


Figure 154. The Trst Reset Diagram in OCDS_DEBUGPORT

b) Hardware Reset

The external rst signal should be active (high level) for at least two periods of clk clock to reset flip-flops in the clk clock domain.

The external trst signal should be active (low level) for at least one period of tck clock to reset TAP placed in the OCDS_DEBUGPORT unit.

5.20.9 OCDS_DEBUGPORT

a) OCDS_DEBUGPORT Overviev

The OCDS_DEBUGPORT is used for JTAG communication with host controller. This unit contains instruction register, all data registers and NEXUS function controller.

b) OCDS_DEBUGPORT Block Diagram

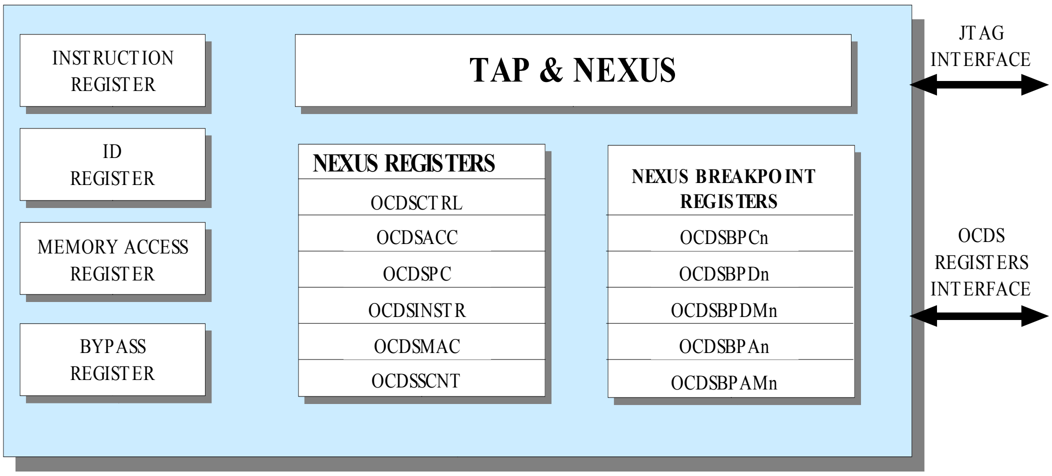


Figure 155. OCDS_DEBUGPORT Block Diagram

c) OCDS_DEBUGPORT Pin Description

Table 135 OCDS_DEBUGPORT Pin Description

Name	Type	Polarity Bus size	Description
		JTAG interface	
tck	I	Rise	**Test Clock** Test clock signal (JTAG 1149.1). Clock pulse for all internal synchronous circuits in TCK clock domain
tms	I	High	**Test mode select** Test mode signal (JTAG 1149).
tdi	I	High	**Test data input** Test data input (JTAG 1149.1).
trst	I	Low	**Test reset** Test reset signal (JTAG 1149).
tdo	O	High	**Test data output** Test data output (JTAG 1149.1).
tdoenable	O	High	**External tristate buffer** Control signal for external tdo tristate buffer.
		CPU interface	
debugrst	O	High	**OCDS reset** Reset signal connected to the CPU reset input port. This signal is logic sum of ocdsen and rst bit from OCDSCTRL register.
		CPU and OCDS_UNIT interface	
updatememaccreg	O	High	**Update memory** Signal causing memory write operation in the OCDS_UNIT. The OCDS_UNIT takes in consideration control information from OCDSMAC register.

Name	Type	Polarity Bus size	Description
capturememaccreg	O	High	**Capture memory** Signal causing memory read operation in the OCDS_UNIT. The OCDS_UNIT takes in consideration control information from OCDSMAC register.
Parallel inputs and outputs to/from registers			
ocdsctrli	I	OCDSCTRL_LEN	**OCDS control register input** OCDS status and control input signal.
ocdsctrlo	O	OCDSCTRL_LEN	**OCDS control register output** OCDS status and control output signal.
ocdsacci	I	OCDSACC_LEN	**Accumulator register input** Accumulator contents.
ocdsdr0i	I	OCDSDR0_LEN	**DR0 register input** DR0 contents.
ocdspci	I	OCDSPC_LEN	**Program counter register input** Program counter contents.
ocdsinstro	O	OCDSINSTR_LEN	**OCDS instruction register output** OCDS instruction register contents.
ocdsmaco	O	OCDSMAC_LEN	**OCDS memory access control register output** OCDS memory access control register contents.
ocdsbpdo	O	(OCDS_NO_OF_BP* OCDSBPD_LEN)	**OCDS breakpoint data registers outputs** OCDS breakpoint data registers contents.
ocdsbpdmo	O	(OCDS_NO_OF_BP* OCDSBPD_LEN)	**OCDS breakpoint data mask register outputs** OCDS breakpoint data mask registers contents.
ocdsbpao	O	(OCDS_NO_OF_BP* OCDSBPA_LEN)	**OCDS breakpoint start address registers outputs** OCDS breakpoint start address registers contents.
ocdsbpamo	O	(OCDS_NO_OF_BP* OCDSBPA_LEN)	**OCDS breakpoint end address registers outputs** OCDS breakpoint end address registers contents.
ocdsbpctrli	I	(OCDS_NO_OF_BP* OCDSBPCTRL_LEN)	**OCDS breakpoint control registers inputs** Current OCDS breakpoint status and control bits.
ocdsbpctrlo	O	(OCDS_NO_OF_BP* OCDSBPCTRL_LEN)	**OCDS breakpoint control registers outputs** OCDS breakpoint control registers contents.

Name	Type	Polarity Bus size	Description
ocdsmemaccregi	I	OCDSMEMACCREG_LEN	**OCDS memory access data register input** Value which will be loaded into OCDS memory access data register (current accumulator contents).
ocdsmemaccrego	O	OCDSMEMACCREG_LEN	**OCDS memory access data register output** OCDS memory access data register contents.
Inputs and outputs designed for Trace (when OCDS_TYPE parameter is 1 or 2 only)			
jbuf_reg	I	OCDSBUFF_LEN	**Buffer Register Input**
capturedr	O	High	**Data Capture Register**
bufenable	O	High	**Trace Buffer Enable**
ocdstrci	I	OCDSTRC_LEN	**OCDS trace control register input** Trace control register (OCDSTRC) contents (status flags).
ocdstrco	O	OCDSTRC_LEN	**OCDS trace control register output** Trace control register (OCDSTRC) contents (control flags).

d) TAP Controller Description

TAP controller is designed according to the state diagram shown in Figure 148.

e) Shift Registers

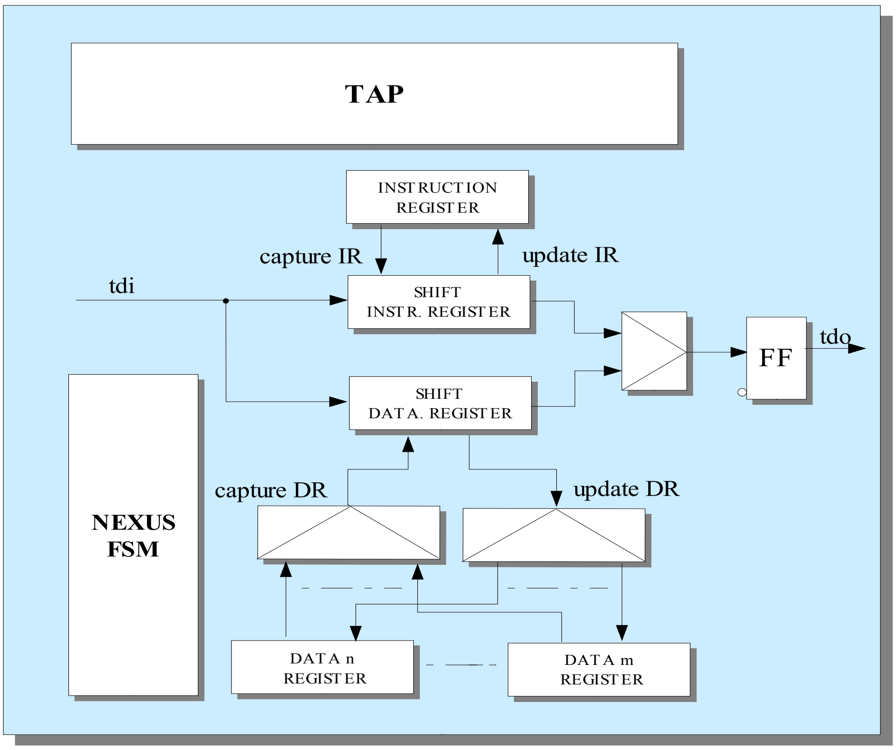


Figure 156. Data Capture and Data Update Solution

The OCDS_DEBUGPORT contains two shift registers: one for the instruction scan and second for the data scan.

The shift instruction register captures decimal value 1 on the rising tck edge in CAPTURE IR state (refer to Figure 156). Instruction is shifted into shift register from tdi input on the rising tck edge. On tdo output captured value appears bit by bit. in UPDATE IR state new instruction is written into the instruction register.

The shift data register captures proper value from active data register (the register depends on the current instruction) through multiplexer on the rising edge of tck in CAPTURE DR state (refer to Figure 156). Data is shifted into the shift register from tdi input on the rising tck edge. On tdo output captured value appears bit by bit. in UPDATE DR state new data is written into proper data register through demultiplexer.

Figure 157 presents the instruction shift register implementation. in CAPTURE IR STATE decimal value 1 is captured into the instruction shift register. The LSB bit (tdoi) is written into tdo output register on the falling tck edge (if instruction scan is selected). Register is shifted on the rising tck clock edge. Input tdi value is written into highest bit of instruction shift register.

In UPDATE IR state value from shift register is written into instruction register. Instruction shift register size is equal to instruction code size.

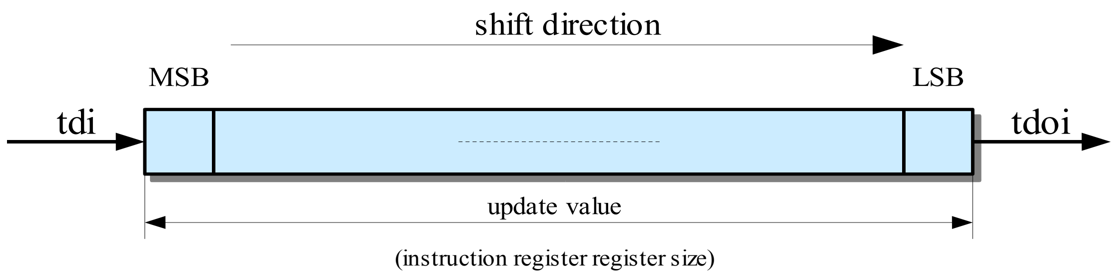


Figure 157. Instruction Shift Register Implementation

Figure 158 presents the data shift register implementation. in CAPTURE DR state value from selected register is captured into lowest bits of shift register and the remaining bits are written with 0. LSB bit (tdod) is written into tdo output register on the falling tck edge (if data scan is selected). Register is shifted on the rising tck clock edge. Input tdi value is written into the highest bit of the shift register. The only exception of this is BYPASS register. in this case tdi input value is written into the lowest bit of the shift register.

In UPDATE DR STATE value from highest bits of shift register are written into selected data register. Data shift register size is equal to the longest data register size. in case of capturing or updating registers which are shorter then shift register only valid number of bits are taken into consideration.

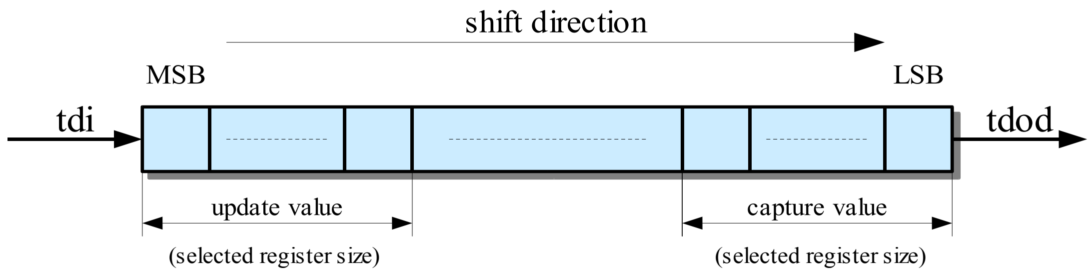


Figure 158. Data Shift Register Implementation

Current active scan (data or instruction) decides which shift register is connected to the tdo output.

f) Updating and Capturing Data in Memory Access

If OCDSMEMACC instruction is chosen:

- in CAPTURE DR STATE capturememaccreg signal is generated
- in UPDATE DR STATE updatememaccreg is generated.

The signals mentioned above and configuration information from OCDSMAC register are used in the OCDS_UNIT for memory access operation.

g) Nexus Access

To enable NEXUS functions Nexus-enable instruction must be written in to JTAG instruction register. Relations between JTAG and NEXUS controller are shown in Figure 147.

Internal logic decodes kind of operation (read or write) and Nexus register address. First step after enabling Nexus controller is to write an input command (Table 117) through data scan. After kind of operation and register address detection – proper register is enabled (each Nexus register has its own enable signal). in the second pass through the data scan write to or read from addressed register is done.

For detailed information about accessing Nexus registers refer to 5.20.4 b).

5.20.10 OCDS_UNIT

a) OCDS_UNIT Overview

Main tasks for the OCDS_UNIT are:

- halting and starting CPU (upon given conditions),
- memory read/write access operations in debug mode,
- instruction substitution in step mode (instruction can be executed from code memory or OCDS instruction register),
- forming status bits placed in OCDS registers.

b) OCDS_UNIT Block Diagram

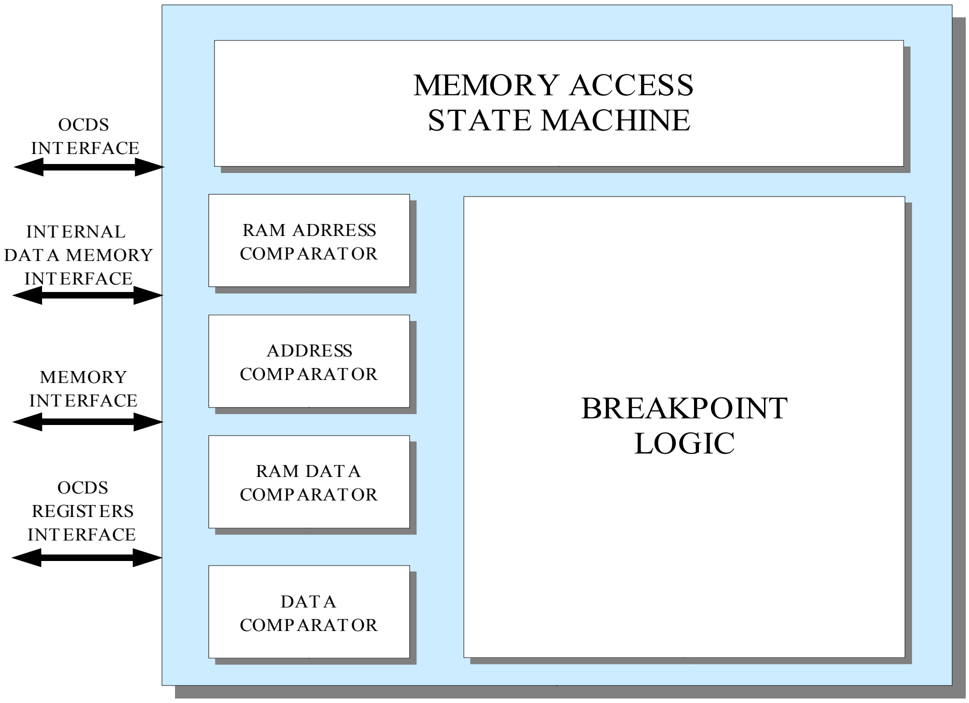


Figure 159. OCDS_UNIT Block Diagram

c) OCDS_UNIT Pin Description

Table 136. OCDS_UNIT Pin Description

Name	Type	Polarity Bus size	Description
clk	I	High	**Clock** Pulse for all internal synchronous circuits.
rst	I	High	**Reset input** External reset signal.
debugrst	O	High	**Reset output** Internal reset signal from debugger.
ocdsupdatememr	I	High	**Update memory** Signal causing memory write operation in the OCDS Unit. The OCDS Unit takes in consideration control information from OCDSMAC.
ocdscapturememr	I	High	**Capture memory** Signal causing memory read operation in the OCDS Unit. The OCDS Unit takes in consideration control information from OCDSMAC.
debugack	I	High	**Debug mode acknowledge** Debug acknowledge signal from CPU.
flush	I	High	**Flush** Program branch indicator.
codefetche	I	High	**Code fetch** Code fetch indicator.

Name	Type	Polarity Bus size	Description
datafetche	I	High	**Data fetch** Data fetch indicator.
lastcycle	I	High	**Last cycle of instruction** Last cycle of instruction indicator.
allfetch	I	High	**All instruction fetch** All instruction fetch (performed and annuled indicator.
debugreq	O	High	**Debug mode request** Debug mode request signal to CPU.
debugstep	O	High	**Debug step** Debug step signal (single instruction execution).
debugprog	O	High	**Debugger program** Instruction fetch source select signal.
datao	I	8	**External memory data bus input** Memory data bus output from CPU, connected to external data memory input. Combinatorial input signal for OCDS_UNIT unit used for breakpoint detection.
memaddr	I	16	**External memory address bus** External memory address bus output from CPU, connected to external data memory input. Combinatorial input signal for OCDS_UNIT unit used for breakpoint detection.
memwr	I	High	**External memory write enable** External memory write signal output from CPU (memwr), connected to OCDS_UNIT input. Combinatorial input signal.
memrd	I	High	**External memory read enable** External memory read signal output from CPU (memrd), connected to OCDS_UNIT input. Combinatorial input signal.
waitstaten	I	Low	**Waitstate indicator** Active low when the CPU performs a wait cycle (internally or externally generated)
memdatai	O	8	**CPU memory data bus input** Memory data bus output from OCDS Unit, connected to CPU (memdatai input port). Combinatorial output signal driven with either external data memory output bus or substituted instruction (in step mode or memory access).
debugmemdatai	I	8	**External memory data bus output** Memory data bus output from external memory, connected to OCDS_UNIT input. Combinatorial input signal.

Name	Type	Polarity Bus size	Description
ramdatai	I	8	**Internal RAM data output** Internal RAM data output bus (from internal RAM to processor) connected to OCDS Unit for breakpoint detection. Combinatorial input signal.
ramaddr	I	8	**Internal RAM address bus** Internal RAM address bus connected to OCDS Unit for breakpoint detection. Combinatorial input signal.
ramwe	I	High	**Internal RAM write enable** Internal RAM write enable signal connected to OCDS Unit for breakpoint detection. Combinatorial input signal.
ramoe	I	High	**Internal RAM output enable** Internal RAM read enable signal connected to OCDS Unit for breakpoint detection. Combinatorial input signal.
ocdsen	I	High	**OCDS enable** This signal enables OCDS functions.
dreq	I	High	**Debug request** Debug request bit from OCDS control register.
dstep	I	High	**Debug step** Debug step request from OCDS control register.
dprog0	I	High	**Debugger program** Fetch source selector (user program memory or OCDS instruction register) bits from OCDS control register.
dprog1	I	High	
dack	O	High	**Debug acknowledge** Debug acknowledge signal from CPU.
swbstop	O	High	**Software breakpoint** Software breakpoint status.
ocdsinstr	I	OCDSINSTR_LEN	**OCDS instruction input** Contents of OCDS instruction register.
ocdsrd	I	High	**Memory read** Read from program/data memory (depends on daprosel signal).
ocdswr	I	High	**Memory write** Write to program/data memory (depends on daprosel signal).
daprosel	I	High	**Data/program memory select** Memory selector signal.
regfilesel	I	High	**Data/program memory select** Memory selector signal.
ocdsbpdvecto	I	(OCDS_NO_OF_BP* OCDSBPD_LEN)	**OCDS breakpoints data input** Contents of all OCDS breakpoint data registers.

Name	Type	Polarity Bus size	Description
ocdsbpdmvecto	I	(OCDS_NO_OF_BP* OCDSBPD_LEN)	**OCDS breakpoints data mask input** Contents of all OCDS breakpoint data mask registers.
ocdsbpavecto	I	(OCDS_NO_OF_BP* OCDSBPA_LEN)	**OCDS breakpoints start address input** Contents of all OCDS breakpoint start address registers.
ocdsbpamvecto	I	(OCDS_NO_OF_BP* OCDSBPA_LEN)	**OCDS breakpoints end address input** Contents of all OCDS breakpoint end address registers.
ocdsbpcvecto	I	(OCDS_NO_OF_BP* OCDSBCTRL_LEN)	**OCDS breakpoints control input** Contents of all OCDS breakpoint control registers.
ocdsbpcvecti	O	(OCDS_NO_OF_BP* OCDSBCTRL_LEN)	**OCDS breakpoints control output** Breakpoint status which will be updated in breakpoint control registers.
ocdsmemaccrego	I	OCDSMEMACCREG_LEN	**OCDS memory access data input** Contents of all OCDS memory access data register.
Output designed for Program Memory Write			
debugpmw	O	High	**OCDS program memory write** Program memory write signal for debugger using (memory write).
Inputs and outputs designed for PMU (when PMU_IMPLEMENT = 1 only)			
debugperen	O	High	**Peripherals clock enable** Peripheral clock enable signal.
clkperen0	I	High	**Peripheral clock enable** This signals enables clock signal for peripherals in debug mode.
clkperen1	I	High	
Inputs and outputs designed for Trace (when OCDS_TYPE parameter is 1 or 2 only)			
full_bp_en	I	High	**Trace breakpoint enable** Breakpoint enable if trace buffer is full (control flag).
full	I	High	**Trace buffer full** Trace buffer full indicator.
cdfetch_t	O	High	**Code or data fetch** Code or data fetch indicator.
user_prog_t	O	High	**User program** User program indicator.
bp_stopi	I	OCDS_NO_OF_BP	**Breakpoint** Breakpoint detection signal.
hit_coderd	O	OCDS_NO_OF_BP	**Code memory read event** Code memory read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).

Name	Type	Polarity Bus size	Description
hit_codefe	O	OCDS_NO_OF_BP	**Instruction code fetch event** Instruction code read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit_codewr	O	OCDS_NO_OF_BP	**Code memory write event** Code memory write event indicator. Write event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit_xramwr	O	OCDS_NO_OF_BP	**External data memory write event** External data memory write event indicator. Write event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit _xramrd	O	OCDS_NO_OF_BP	**External data memory read event** External data memory read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit _iramwr	O	OCDS_NO_OF_BP	**Internal data memory write event** Internal data memory write event indicator. Write event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit _iramrd	O	OCDS_NO_OF_BP	**Internal data memory read event** Internal data memory read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).

5.20.11 OCDS_TRACE Unit

a) Trace Overview

TRACE is used for program and data trace frames collecting. This unit contains trace buffer controller.

b) Trace Block Diagram

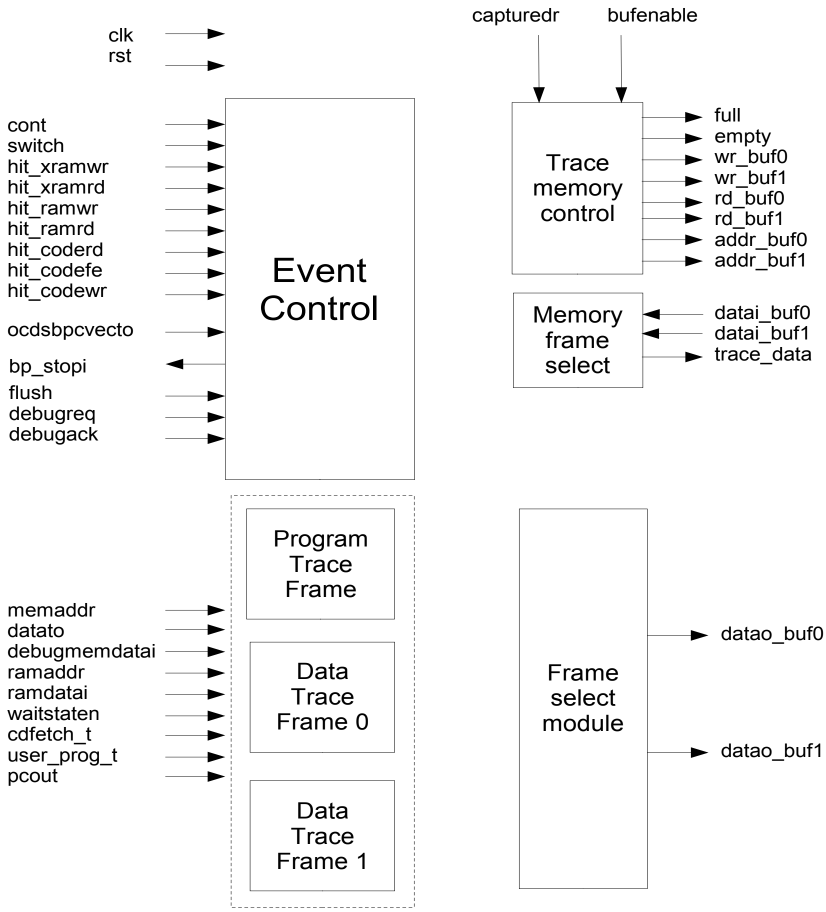


Figure 160. OCDS_TRACE Block Diagram

Event Control

Event Control block is responsible for triggering trace frames generation. It starts and stops trace frames collecting.

Program Trace Frame, Data Trace Frame 0, Data Trace Frame 1

These blocks are responsible for generation of particular fields in trace frames.

Frame select module

Formed trace frames are passed to the trace buffer by this module.

Trace memory control

Trace memory is FILO structure. This unit controls read/write operations from/to the trace buffer. The Trace memory control generates full and empty trace buffer flags.

Memory frame select

Trace frames are placed in the trace buffer. The trace buffer is made as two RAM memories. The Memory frame select module selects trace frame from one of these memories and passes it into trace_data output port.

c) Trace Pin Description

Table 137. OCDS_TRACE Pin Description

Name	Type	Polarity Bus size	Description
clk	I	Rise	**Clock** Pulse for all internal synchronous circuits.
rst	I	High	**Reset** External reset signal for CPU.
codefetch	I	High	**Code fetch** Code fetch indicator.
flush	I	High	**Flush** Code fetch after branch indicator.
pcout	I	24	**Program counter** Contents of the CPU program counter register.
capturedr	I	High	**Capture data scan** Capture data scan state indicator.
bufenable	I	High	**Trace buffer enable** Trace buffer enable signal.
debugreq	I	High	**Debug request** Debug request signal.
debugack	I	High	**Debug acknowledge** Debug mode acknowledge signal.
trace_data	O	BUF_LEN	**Trace data** Trace data bus contains read frame from the trace buffer.
full	O	High	**Trace buffer full** Trace buffer full indicator.
empty	O	High	**Trace buffer empty** Trace buffer empty indicator.
cont	I	High	**Continue bit**
switch*	I	High	**Switch trace bit**
memaddr*	I	24	**External memory address bus** External memory address bus output from CPU, connected to external data memory input.
datao*	I	8	**External memory data bus input** Memory data bus output from CPU, connected to external data and program memory input.
debugmemdatai*	I	8	**External memory data bus output** Memory data bus connected to CPU memdatai input.
ramaddr*	I	8	**Internal RAM address bus** Internal RAM address bus.
ramdatai*	I	8	**Internal RAM data output** Internal RAM data output bus (from internal RAM to processor).

Name	Type	Polarity Bus size	Description
waitstaten	I	High	**Waitstate indicator** Active low when the CPU performs a wait cycle (internally or externally generated)
cdfetch_t	I	High	**Code or data fetch** Code or data fetch indicator.
user_prog_t	I	High	**User program** User program indicator.
bp_stopi	O	OCDS_NO_OF_BP	**Breakpoint** Breakpoint detection signal.
ocdsbpcvecto	I	OCDS_NO_OF_BP* OCDSBPCTRL_LEN	**OCDS breakpoints control input** Contents of all OCDS breakpoint control registers.
hit_coderd	I	OCDS_NO_OF_BP	**Code memory read event** Code memory read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit_codefe	I	OCDS_NO_OF_BP	**Instruction code fetch event** Instruction code read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit_codewr	I	OCDS_NO_OF_BP	**Code memory write event** Code memory write event indicator. Write event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit_xramwr	I	OCDS_NO_OF_BP	**External data memory write event** External data memory write event indicator. Write event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit _xramrd	I	OCDS_NO_OF_BP	**External data memory read event** External data memory read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit _iramwr	I	OCDS_NO_OF_BP	**Internal data memory write event** Internal data memory write event indicator. Write event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).
hit _iramrd	I	OCDS_NO_OF_BP	**Internal data memory read event** Internal data memory read event indicator. Read event fulfils OCDSBPAn, OCDSBPAMn, OCDSBPDn, OCDSBPDMn and OCDSBPCn (except *mod* field).

Name	Type	Polarity Bus size	Description
addr_buf0	O	BUF_SIZE	**RAM0 address** RAM0 address bus.
datao_buf0	O	BUF_LEN	**RAM0 data output** RAM0 data output bus.
datai_buf0	I	BUF_LEN	**RAM0 data input** RAM0 data input bus.
wr_buf0	O	High	**RAM0 write enable** RAM0 write enable signal.
rd_buf0	O	High	**RAM0 read enable** RAM0 read enable signal.
addr_buf1	O	BUF_SIZE	**RAM0 address** RAM1 address bus.
datao_buf1	O	BUF_LEN	**RAM1 data output** RAM1 data output bus.
datai_buf1	I	BUF_LEN	**RAM1 data input** RAM1 data input bus.
wr_buf1	O	High	**RAM1 write enable** RAM1 write enable signal.
rd_buf1	O	High	**RAM1 read enable** RAM1 read enable signal.
***** - *PORTS exists only in configuration with data trace (OCDS_TYPE = 2).*			

d) Trace Description

TRACE unit monitors program flow and read/write operations in the CPU in normal program execution mode. According to trace configuration trace frames are written into the trace buffer. The trace buffer is read in the debug mode through JTAG interface.

There are two kinds of trace:

- Program trace – after detecting any taken branch program trace frame is generated. Trace frame contains information about given branch.
- Data trace – after detecting read/write operation which fulfils data trace attributes (possible attributes are the same as for the breakpoint) data trace frame is generated. Trace frame contains information about given read/write operation.

Possible OCDS configurations:

- OCDS without trace.
- OCDS with program trace.
- OCDS with program and data trace.

e) Starting and Finishing Trace

Trace start - possible methods:

- Breakpoint configuration (start trace program/data on read/write event in the CPU).

- Setting *cont* bit ('0' to'1' in normal program execution mode) in the OCDSCTRL register. Event starts simultaneously program and data trace when *switch* bit is set (OCDSCTRL register). When *switch* is cleared event starts program trace only.
- Leaving debug mode with *cont* bit set to '1'. Event starts simultaneously program and data trace when *switch* bit is set (OCDSCTRL register). When *switch* is cleared event starts program trace only.
- Start trace in debug mode is done by setting *cont* bit from '0' to '1'.

Trace end - possible methods:

- Breakpoint configuration (end program/data trace on read/write event in the CPU).
- Trace (program and data) ends every time when the CPU enters into debug mode (it is possible to enter into the debug mode on trace buffer full). If enabled trace is required after leaving debug mode *cont* and *switch* bits (OCDSCTRL register) must be set before (please refer to trace start possible methods mentioned earlier).

f) Trace Frames

Table 138. Trace Frames

Frame Name	CODE	Description
PT_BR	0000	Program Trace Branch frame is generated when branch occurs.
START_PT_FR	0010	Program Trace Start frame is generated when event which starts program trace does not generate program trace frame (frames with PT prefix) and START_PT_DT_FR frame.
START_ PT_DT_FR	0100	Program and Data Trace Start frame is generated when events which start program and data trace in the same time do not generate program and data trace frame (frames with PT and DT prefixes).
START_ DT_FR	0011	Data Trace Start frame is generated when event which starts data trace does not generate data trace frame (frames with DT prefix) and START_PT_DT_FR frame.
DT_CODE_RD	1000	Data Trace Code Memory Read frame is generated when code memory read operation performed by the CPU fulfills data trace configuration.
DT_CODE_FE	0101	Data Trace Code Fetch frame is generated when the CPU fetches instruction which fulfills data trace configuration.
DT_CODE_WR	1100	Data Trace Code Memory Write frame is generated when code memory write operation performed by the CPU fulfills data trace configuration.
DT_IRAM_RD	1001	Data Trace Internal Data Memory Read frame is generated when internal data memory read operation performed by the CPU fulfills data trace configuration.
DT_IRAM_WR	1101	Data Trace Internal Data Memory Write frame is generated when internal data memory write operation performed by the CPU fulfills data trace configuration.
DT_XRAM_RD	1010	Data Trace External Data Memory Read frame is generated when external data memory read operation performed by the CPU fulfills data trace configuration.

Frame Name	CODE	Description
DT_XRAM_WR	1110	Data Trace External Data Memory Write frame is generated when external data memory write operation performed by the CPU fulfills data trace configuration.
Note 1: Length and value of the CODE field stays unchanged in both possible trace configurations (configuration with program trace and configuration with program and data trace). *Note 2: in trace configuration without data trace PT_BR and START_PT_FR frames are generated only.*		

g) Trace Frames Formats

Format 1

Format 1 is used for following frames:

PT_BR

Table 139. Frame Format 1

Field name	Bit	Description
START_BIT	64	START_BIT=1 when event which started program trace generated data trace frame. START_BIT=0 other cases.
CODE	63..60	Trace frame code (see Table 138).
START_ADDR	59..36	Branch instruction address or instruction address executed just before interrupt call.
END_ADDR	35..12	First instruction address after branch or first instruction address in interrupt handler.
UNB	11..0	Unused bits read as 0.

Format 2

Format 2 is used for following frames:

PT_BR_EX

Table 140. Frame Format 2

Field name	Bit	Description
START_BIT	64	START_BIT=1 when event which started program trace generated data trace frame. START_BIT=0 other cases.
CODE	63..60	Trace frame code (see Table 138).
START_ADDR	59..36	Branch instruction address or instruction address executed just before interrupt call.
END_ADDR	35..12	First instruction address after branch or first instruction address in interrupt handler.
DATA	11..4	Trace data from code memory read operation. This field is complementary information for data trace.
UNB	3..0	Unused bits read as 0.

Format 3

Format 3 is used for following frames:

DT_CODE_RD

DT_CODE_WR

DT_IRAM_RD

DT_IRAM_WR

DT_XRAM_RD

DT_XRAM_WR

Table 141. Frame Format 3

Field name	Bit	Description
START_BIT	64	START_BIT=1 when event which started program trace generated data trace frame. START_BIT=0 other cases.
CODE	63..60	Trace frame code (see Table 138).
PC	59..36	Address of instruction which performed read/write operation. This field is not valid for DT_CODE_FE frame. Fetched instruction address is given in DATA_ADDR field.
DATA_ADDR	35..12	Address of data in read/write operation.
DATA	11..4	Data which was read from/write to memory in read/write operation.
UNB	3..0	Unused bits read as 0.

Format 4

Format 4 is used for following frames:

START_PT_FR

START_PT_DT_FR

START_DT_FR

Table 142. Frame Format 4

Field name	Bit	Description
START_BIT	64	START_BIT=1.
CODE	63..60	Trace frame code (see Table 138).
START_ADDR	59..36	Address of currently executed instruction except situation when trace starts on instruction fetch then START_ADDR equals to fetched instruction address.
UNB	35..0	Unused bits read as 0.

- Format 5

Format 5 is used for following frames:

DT_CODE_FE

Table 143. Frame Format 5

Field name	Bit	Description
START_BIT	64	START_BIT=1.
CODE	63..60	Trace frame code (see Table 138).
START_ADDR	59..36	Address of instruction from which any of the bytes caused data read match.
INSTR	35..4	Top of instruction queue, consisting of the actual instruction and eventually next instruction data, too.
VALID	3..0	Four bytes of instruction validity bits; when a bit is 1, it means the corresponding byte of 'INSTR' has met the data read match. Bit 0 corresponds with field [11..3] and bit 3 with [35:28].

The DT_CODE_FE frame is generated at the end of the execution of the instruction that is aligned in to the high end of the INSTR field. As there can be multiple bytes "taken" by the CPU in a single clock cycle, this frame only provides the address of the 1st byte of an instruction and four indicators pointing which of the maximum four bytes of instruction caused a data trace match.

Start bit is set in trace frames or start frame is generated every time:

- On leaving debug mode with enabled trace (*cont* bit in the OCDSCTRL register is set).
- On start trace event (set in OCDSBP* registers).
- When change of *cont* bit (OCDSCTRL register) from '0' to '1' in normal execution mode is detected.

h) The Trace Buffer

The trace buffer is FILO (First in Last Out) structure. Trace frames generated during instruction execution are written into the buffer. Physically the buffer is built as two RAM memories. Those memories can be written in the same time with two different frames. One frame occupies one RAM location.

The frames are written into the memory on the rising clkcpu edge (in normal CPU mode) when write enable signal is active. Read signal is asserted in tck clock domain but the buffer is read on the rising clkcpu edge when read falling edge is detected (tck clock period equals at least two clkcpu periods). Read signal is active when JTAG controller is in the CAPTURE_DR state and the BUFF instruction is the JTAG instruction register.

i) Trace Buffer Full Flag

- The *full* buffer flag is placed in OCDSCTRL register. The flag is set when in the trace buffer is space for 3 trace frames only. The flag is set to '0' when all frames are read from the trace buffer or when the trace buffer is cleared (changing the JTAG BUFF instruction into any other JTAG instruction).
-

When the trace buffer is full it is possible:

- To stop program execution and enter into the debug mode when *full_bp_en* bit is set to '1' in OCDSCTRL register.
- To stop trace frames collecting without stopping program execution when *full_bp_en* bit is set to '0' in OCDSCTRL register. Collected frames are not overwritten.

j) Trace Buffer Empty Flag

The *empty* trace buffer flag is placed in JTAG data BUFF register. The flag is set to '1' when the trace buffer do not contain trace frames. Trace frames read from the BUFF register with empty *flag* set to '1' are not valid. The flag is set to '0' when the trace buffer contains at least one trace frame.

k) Trace Buffer Read Rules

- Trace buffer read is performed only in the debug mode through the JTAG interface until empty *flag* in the BUFF register is set.
- Values read from the BUFF register with empty *flag* set to '1' are not valid.

- Frames are read from trace buffer in FILO-like order (the last generated frames are read first).
- The trace buffer is cleared after changing the JTAG BUFF instruction into any other JTAG instruction.
- After updating the JTAG instruction register with the BUFF instruction internal buffer pointer is decremented if in the trace buffer exist at least one free memory location for trace frame.

l) Trace in Debug Mode

It is possible to generate trace frames in debug mode if trace is enabled.

Trace in debug mode rules:

- start program/data trace frames are not generated.
- START_BIT is set to 0 in frames generated by step from code memory.
- START_BIT is set to 1 in frames generated by step from OCDSINSTR register and memory access operations (through OCDSMAC register).

5.20.12 Parameters

Table 144 OCDS Parameters

Name	Type	Default value	Description
Main parameters			
OCDS_TYPE		0	
OCDS_NO_OF_BP	Integer	4	Number of breakpoints.
Register addresses			
MAX_REG_LEN	Integer	46	Maximum data register size.
ADR_NX_VND_REG_AREA	Integer	64	Begin of vendor defined registers in NEXUS address area.
OCDSCTRL_ADR	Integer	0	OCDSCTRL register address given relatively to ADR_NX_VND_REG_AREA.
OCDSACC_ADR	Integer	1	OCDSACC register address given relatively to ADR_NX_VND_REG_AREA.
OCDSPC_ADR	Integer	2	OCDSPC register address given relatively to ADR_NX_VND_REG_AREA.
OCDSINSTR_ADR	Integer	3	OCDSINSTR register address given relatively to ADR_NX_VND_REG_AREA
OCDSMAC_ADR	Integer	4	OCDSMAC register address given relatively to ADR_NX_VND_REG_AREA.
OCDSTRC_ADR	Integer	5	OCDSTRC register address given relatively to ADR_NX_VND_REG_AREA.

Name	Type	Default value	Description
OCDSBPD1_ADR	Integer	6	OCDSBPD1 register address given relatively to ADR_NX_VND_REG_AREA.
OCDSBPDM1_ADR	Integer	7	OCDSBPDM1 register address given relatively to ADR_NX_VND_REG_AREA.
OCDSBPA1_ADR	Integer	8	OCDSBPA1 register address given relatively to ADR_NX_VND_REG_AREA.
OCDSBPAM1_ADR	Integer	9	OCDSBPAM1 register address given relatively to ADR_NX_VND_REG_AREA.
OCDSBPCTRL1_ADR	Integer	10	OCDSBPC1 register address given relatively to ADR_NX_VND_REG_AREA.
TAP parameters			
IR_DEF_LEN	Integer	4	Instruction register size.
ID_CODE_LEN	Integer	32	ID Register size.
ID_CODE_VALUE	Bit vector	50000321h	Contents of the ID register. Refer to ID_CODE_LEN.
Instruction codes			
IDREG_IR	Bit vector	1h	Identification instruction code. Refer to IR_DEF_LEN.
NXENABLE_IR	Bit vector	Bh	Enable Nexus controller instruction code. Refer to IR_DEF_LEN.
MEMACCREG_IR	Bit vector	Ah	Memory access instruction code. Refer to IR_DEF_LEN.
BYPAS_IR	Bit vector	Fh	Bypass instruction code. Refer to IR_DEF_LEN.
BUFFREG_IR	Bit vector	9h	Trace Memory Access Register Enable
Note: OCDSBPD1_ADR, OCDSBPDM1_ADR, OCDSBPA1_ADR, OCDSBPAM1_ADR, OCDSBPC1_ADR addresses are relative to ADR_NX_VND_REG_AREA. Mentioned addresses are correct for the first set of breakpoint registers. to calculate addresses for the following breakpoint registers it is necessary to add multiple of 5 (5 for second breakpoint, 10 for third breakpoint registers etc) for proper register base address calculation (for default values refer to Table 121*).*			
Examples: *Address of OCDSBPA2 register for second breakpoint is equal:* **ADR_NX_VND_REG_AREA + OCDSBPA1_ADR + 5** *Address of OCDSBPC4 register for fourth breakpoint is equal:* **ADR_NX_VND_REG_AREA + OCDSBPC1_ADR + 15**			

5.21. SOFTRSTCTRL

5.21.1 Overview

The SOFTRSTCTRL component implements functions which allow to reset all components of the R80251XC core (except the SOFTRSTCTRL core). The software reset is performed by SFR write sequence.

5.21.2 Pin Description

Table 145 SOFTWARE RESET Pin Description

Name	Type	Polarity Bus size	Description
clkcpu	I	Rise	**Clock** Clock input for all internal synchronous logic
resetff	I	High	**Internally generated reset:** Used to reset all internal synchronous logic
newinstr	I	High	**New instruction indicator** Indicates the previous cycle was the 1st cycle of a new instruction
debugack	I	High	**Debug acknowledge** Indicates that the CPU has stopped instruction execution and entered the Debug Mode
watchdogstop	I	High	**Watchdog stop** Used only in the Debug Mode, is active when the CPU is not executing any instruction from normal program
srstreq	O	High	**Software reset request** Software reset request – system reset is performed.
srstflag	O	High	**Software reset flag** Software reset status flag output.
sfrdatai	I	8	**Special Function Registers Data Bus input** Contains data written to Special Function Registers
sfraddr	I	7	**SFR register Address bus** Address of SFR register that CPU wants to read from or write to.
sfrwe	I	High	**SFR Write Enable** Indicates a write access to the Special Function Register

5.21.3 Description

A software reset will be generated after two consecutive '1' value writes to the "srstreq" bit in the "srst" register (0F7h). This solution will prevent accidental reset evoking.

It will be possible to identify if the source of the reset sequence was a software reset, by reading the "srstreq" bit in the "srst" register.

A software reset won't have influence on the debugger.

A software reset will be visible outside of the core by the "ro" output (this output will be logic 'OR' of watchdog, debugger and software reset signals).

5.22. RTC

5.22.1 Overwiew

The Real Time Clock realizes a real time count with a resolution of 1/256th second. It allows the user to read seconds, minutes, hours, day of the week and the date. The date is represented by a 16-bit number, which value is interpreted by the user software. The RTC enables a count of 179 years. The RTC features an alarm function which may be used to generate interrupts at a given time during a day or periodically. These interrupts may be used to resume operation from the IDLE/STOP mode at a given time.

5.22.2 Block Diagram

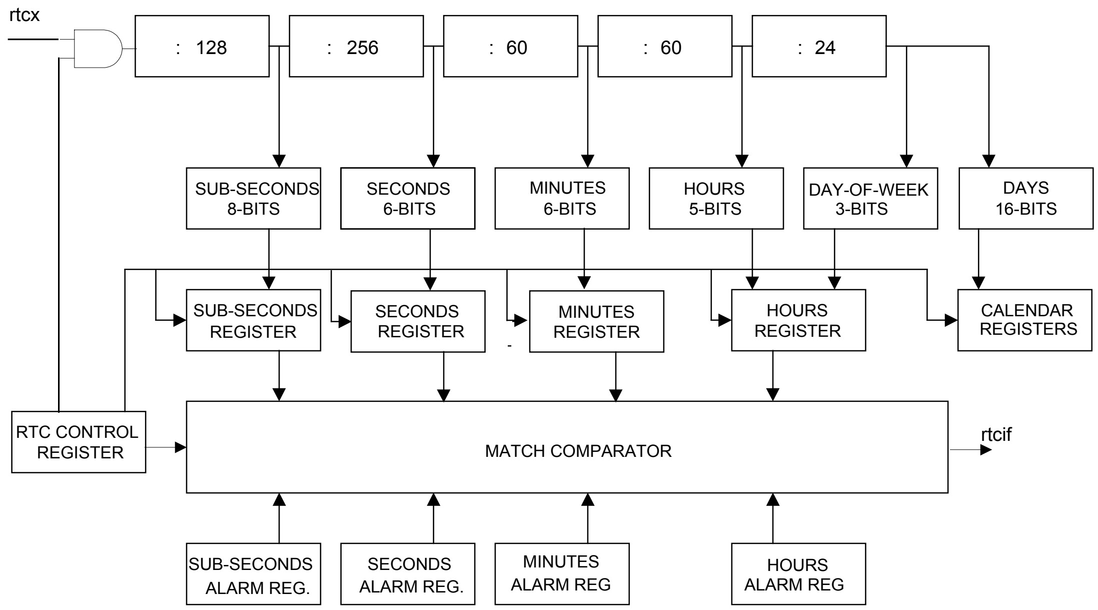


Figure 161. RTC Block Diagram

5.22.3 Pin Description

Table 146 RTC Pin Description

Name	Type	Polarity Bus size	Description
clk	I	Rise	R80251XC Peripheral Clock input
rst	I	High	Internal synchronous reset
rtcreset	I	High	RTC asynchronous reset
rtcx	I	Rise/Fall	RTC Clock input
rtcifo	O	High	RTC Interrupt Flag to the Interrupt Controller
rtcifa	O	High	RTC Interrupt Flag to the Power Management Unit
sfrdatai	I	8	SFR data bus input
rtcsel	O	4	RTC Select Register output
rtcdata	O	8	RTC Data Register output
sfraddr	I	7	SFR address bus input
sfrwe	I	High	SFR write enable input

5.22.4 Description

a) Starting and Stopping the RTC

Starting and stopping the RTC is controlled with rtce bit (rtcc.0). When the bit is set all the registers triggered with the RTC clock (rtcx) are enabled and the RTC counters are clocked. When this bit is cleared the state of the registers freezes. The value of the clock registers will be preserved along with all control and alarm settings.

b) Setting and Reading the RTC Registers

Access to the RTC registers (rtcss, rtcs, rtcm, rtch, rtcd0, rtcd1) is controlled with the rtcwe and rtcre bits. to enable reading the RTC registers, the rtcre bit must be set. to enable writing the RTC registers, the rtcwe bit must be set. Updates to the time keeping registers are suspended until rtcwe (or rtcre) is cleared. Any access (reading or updating) should be accomplished within 1 ms from an appropriate bit setting. If a subsecond timer tick occurs during the access window it will be processed as soon as rtcwe (or rtcre) is cleared.

The rtcwe and rtcre bits are not allowed to be set simultaneously. Attempt of doing so will be ignored. If not cleared in the software within 1.95 ms after being set each of these bits will be cleared automatically.

Writing an invalid time (i.e. a value outside the range representing valid second, minute or hour value) to the RTC registers will result in an inaccurate count by RTC.

c) Using the RTC Alarm

The alarm function is used to generate an interrupt when the RTC value matches a value from the enabled alarm register. When the RTC reaches the selected alarm value, it sets a flag (rtcif). This will cause an interrupt if enabled, even in Stop mode. With interrupt not enabled, it can still wake-up the R80251XC from IDLE or STOM mode.

The user selects the alarm time by setting the rtass, rtas, rtam and rtah registers. It is not necessary to set the rtcwe or rtcre bits to access alarm registers. There is no specific alarm register bound to the RTC Day count or RTC Day of Week register. Therefore it is up to user software to control the execution of an alarm that is desired only at a specific date. Writing an invalid value (e.g. 61 to rtam or rtah registers) will never cause a match. It is the responsibility of the user to ensure that only valid time values are written to alarm registers.

The alarm function is implemented with a comparator that matches the alarm value set by the user against the current RTC value. The user can select a match of one or more of the following registers: rtass, rtas, rtam and rtah against their RTC counterparts by setting appropriate compare enable bits (rtcc.7-4). Any RTC register that has a corresponding compare enable bit set to 0 is treated as a match. This allows an alarm to occur once a day at specific point in time (when all compare enable bits are set) or periodically - once per second, once per minute, once per hour within a given period of time (depending on a combination of the compare enable bits that are set).

The general procedure for setting the alarm registers to cause an RTC interrupt is as follows:

- clear the ertc bit of the ien4 register and all alarm compare enable bits (rtcc.7-4),
- write one or more alarm registers (rtass, rtas, rtam, rtah),
- set the desired alarm compare enable bits and the ertc bit (ien4.5).

d) RTC Special Function Registers

Real Time Clock is interfaced through 10 special function registers: six that keep time count (rtcss, rtcs, rtcm, rtch, rtcd1, rtcd0) and four that control the alarm time (rtass, rtas, rtam, rtah).

Its operation is controlled by the rtcc register and ertc (ien4.5) bit.

All functions and restrictions on accessing the registers are presented in the “Special Function Registers” section.